Haaga-Helia
ammattikorkeakoulu Oy

Verkkosivujen suunnittelu ja toteutus Vue.js:llä

Tommi Ilvonen

Opinnäytetyö
Tietojenkäsittelyn
koulutusohjelma
2018

Tiivistelmä

Haaga-Helia ammattikorkeakoulu Oy

Tekijä(t)
Tommi Ilvonen.

Koulutusohjelma
Tietojenkäsittelyn

Raportin/Opinnäytetyön nimi	**Sivu- ja liitesivumäärä**
Verkkosivujen suunnittelu ja toteutus Vue.js:llä	31 + 5

Opinnäytetyö käsittelee verkkoportfolion suunnittelua ja toteutusta toimeksiantaja Kalle Pitkäselle. Verkkoportfolio on perinteistä CV:tä nykyaikaisempi tapa esittää osaamista, saavutuksia ja työkokemusta. Verkkoportfolio on käytännössä sama asia kuin yrityksen verkkosivut, mutta brändäyksen ja markkinoinnin kohteena on yrityksen sijasta yksityishenkilö.

Opinnäytetyön toimeksiantajana toimii taiteilija Kalle Pitkänen, joka on suuntautunut öljy- ja temperamaalauksiin. Pitkänen on aiemmin laittanut töitään esille kuvien jakopalvelu Instagramiin. Myyntiä, tavoitettavuutta ja tunnettavuutta edistääkseen Pitkänen on ajatellut tarvitsevansa itselleen verkkoportfolion, johon hän voisi laittaa teoksiaan esille sekä kertoa itsestään.

Opinnäytetyön tavoitteena on toteuttaa toimeksiantajalle verkkoportfolio, joka on vähimmäisvaatimuksiltaan julkaistu, responsiivinen, käytettävä ja ulkoasultaan valmis. Näkymiä pitää olla etusivu, henkilökuvaus, kuvagalleria ja yhteystiedot. Kuvagallerian pitää helppokäyttöinen ja tuoda hyvin esille Pitkäsen taideteokset.

Verkkoportfolio toteutetaan tekijälle ennestään tuntemattomilla web-teknologioilla ja näin tarkoituksena on tukea hänen ammatillista kasvuaan. Opinnäytetyön ei ole tarkoitus kertoa, kuinka tehdä verkkosivut mahdollisimman tehokkaasti ja helposti. Opinnäytetyön tarkoitus on suunnitella ja tuottaa toimeksiantajalle ratkaisu, perustella valitut teknologiat ja kuvata toimeksiannon työvaiheiden kulkua.

Opinnäytetyön teoriaosuus koostuu projektin määrittelystä, jossa käydään läpi erilaisia web-kehitysteknologioita ja toteutustapoja. Määrittelyosuudessa perustellaan ja kuvaillaan valitut teknologiat. Toteutusosuudessa käydään toteutusvaiheittain läpi toteutuksen vaiheita. Opinnäytetyön toteutusaikataulu oli 12.2.2018 – 7.5.2018.

Opinnäytetyön lopputuloksena toimeksiantajalle syntyi verkkosivut kallepitkanen.com. Verkkosivujen sovelluskehyksenä toimi Vue.js ja käyttöliittymäkomponenttikirjastona oli käytössä Vuetify. Verkkosovellus julkaistiin Docker- ja Dokku-teknologioita käyttäen ja julkaisuprosessi automatisoitiin tapahtumaan lyhyellä yhden rivin komennolla.

Asiasanat
Sovelluskehitys, verkkosivut, suunnittelu, määrittely, toteutus, julkaisu

Sisällys

Käytetty sanasto

API (Application Programming Interface) tarkoittaa ohjelmointirajapintaa.
Angular.js on Googlen ylläpitämä avoimen lähdekoodin JavaScript Framework.
CSS (Cascading Style Sheets) on tekniikka, jolla määritellään ulkoasu HTML-kielelle.
CMS (Content Management System) eli sisällönhallintajärjestelmä.
Domain on verkkotunnus eli verkkosivun nimi ja julkinen osoite.
DNS (Domain Name System) on järjestelmä, joka kääntää verkkotunnukset IP-osoitteiksi.
Framework tarkoittaa sovelluskehystä, jota käytetään tuottamaan sovelluksia.
Front-end tarkoittaa päätelaitteella näkyvää osuutta sovelluksesta.
Git on versionhallintaohjelmisto.
HTML (Hypertext Markup Language) on merkintäkieli, jolla kuvataan sivuston rakenne.
JavaScript on ohjelmointikieli, jota käytetään verkkosivujen toiminnallisuuden luomiseen
JavaScript Framework tarkoittaa sovelluskehystä, joka on toteutettu JavaScriptillä.
JSON (JavaScript Object Notation) on yksinkertainen formaatti tiedon välittämiseen.
MPA (Multi-page application) eli perinteinen monisivuinen sovellustoteutus
MVC (Model View Controller) tarkoittaa malli-näkymä-käsittelijä ohjelmistoarkkitehtuuria.
PaaS (Platfrom as a Service) tarkoittaa sovellusalustaa palveluna.
PWA (Progressive Web Application) on yhdistelmä mobiili- ja perinteisestä web-totetusta.
React.js on Facebookin kehittämä JavaScript library.
Repository on versionhallinnan tallennussijainti.
SPA (Single-page application) tarkoittaa yhden sivun sovellustoteutusta.
Vue.js on Evan You:n kehittämä avoimen lähdekoodin JavaScript Framework.
Vuetify on Käyttöliittymäkomponenttikirjasto Vue.js:lle.

1 Johdanto

Opinnäytetyöni käsittelee verkkoportfolion suunnittelua ja toteutusta toimeksiantajalleni. Verkkoportfolio on perinteistä CV:tä nykyaikaisempi tapa esittää osaamista, saavutuksia ja työkokemusta. Verkkoportfolio on käytännössä sama asia, kuin yrityksen verkkosivut, mutta brändäyksen ja markkinoinnin kohteena on yrityksen sijasta yksityishenkilö.

Projektin toimeksiantajana toimii taiteilija Kalle Pitkänen. Hän on öljy- ja temperamaalauksiin suuntautunut taiteilija Sipoosta. Pitkänen on aiemmin laittanut töitään esille kuvien jakopalvelu Instagramiin. Myyntiä, tavoitettavuutta ja tunnettavuutta edistääkseen, Pitkänen on ajatellut tarvitsevansa itselleen verkkoportfolion, johon hän voisi laittaa teoksiaan esille, sekä kertoa itsestään.

Verkkoportfolion tekeminen taitelijalle tarkoittaa, että verkkoportfolion pitää olla persoonallinen ja tuoda taitelijan teokset mahdollisimman hyvin esille. Kuvien selaamisen pitää olla mahdollisimman vaivatonta ja kuvista täytyy saada myös yksityiskohdat esille. Taitelijan persoonaan pitää perehtyä, jotta se saadaan tuotua esille verkkoportfoliossa ja kerrottua kävijöille.

Toimeksiantaja hyötyy verkkoportfoliosta saadessaan teoksilleen sekä itselleen enemmän näkyvyyttä. Parempi näkyvyys helpottaa näyttelyihin pääsemistä sekä töiden myyntiä. Verkkoportfolio antaa henkilöstä ammattimaisen kuvan ja sillä voi erottua edukseen.

1.1 Tavoitteet ja suunnitelma

Tavoitteenani oli tehdä verkkoportfolio itselleni uusilla web-teknologioilla, joten projektilla kasvatin omaa osaamistani. Verkkoportfolio toimii näytteenä omasta osaamisestani. Koska uudella teknologialla tehdessä oppimiskäyrä on alkuun hidas, teknologioiden käyttö pitää suunnitella hyvin ja tavoitteiden pitää olla realistiset.

Toimeksiantaja antoi minulle joustavan aikataulun ja toteutuksen suhteen vapaat kädet. Opinnäytetyön aikataulu oli 12.2.2018 – 7.5.2018, mutta toimeksiantajan kanssa yhteistyö jatkuu ja kehitän verkkosivuja opinnäytetyön jälkeen. Sovimme, että opinnäytetyöprojektin vähimmäisvaatimus on tuottaa verkkoportfolio, jossa on perusasiat kunnossa. Verkkosivujen pitää olla julkisesti saavutettavissa, responsiiviset ja etusivun pitää olla näyttävä. Toimeksiantajan tarvitsemat näkymät ovat etusivu, henkilöesittely, kuvagalleria ja yhteystiedot.

Vähimmäisvaatimuksilla verkkoportfolion ulkonäkö on valmis ja esittelykelpoinen. Kun vähimmäisvaatimukset on toteutettu, seuraavat askeleet olisivat toteuttaa hakukoneoptimointi, ottaa käyttöön sisällönhallintajärjestelmä ja toteuttaa julkaisun automatisointi. Nämä asiat päätettiin jättää opinnäytetyön ulkopuolelle ja toteutan ne sitä mukaa, kun ehdin.

Varsinkin vähimmäisvaatimuksista seuraavat tavoitteet ovat itselleni oppimisen ja ammattitaidon kannalta mielenkiintoisia. Nämä tavoitteet ovat myös toimeksiantajan kannalta tärkeitä verkkosivujen ylläpidon ja tavoitettavuuden osalta.

1.2 Rajaus

Opinnäytetyön ei ole tarkoitus kertoa, kuinka tehdä verkkosivut mahdollisimman tehokkaasti ja helposti. Opinnäytetyössä ei keskitytä kertomaan verkkosivujen historiasta tai asioista, jotka pitää ottaa huomioon verkkosivuja tehdessä. Edellä mainittuja ohjeitä on Internet täynnä, joten ne eivät tuo lisäarvoa tälle opinnäytetyölle.

Projektin tarkoitus on yksinomaan suunnitella ja tuottaa toimeksiantajalle ratkaisu, perustella valitut teknologiat ja kuvata toimeksiannon työvaiheiden kulkua. Olen enemmän ohjelmistokehittäjä, kuin graafikko tai Web Designer, joten lähestymistapani on suunnitella toteutus, toteuttaa ja oppia. En kuitenkaan tingi verkkosivujen ulkonäöstä tai käytettävyydestä. Front-end tarkoittaa selainpuolen ohjelmointia eli koodia, joka ajetaan selaimessa verkkosivua selatessa. Tähän kuuluu esimerkiksi sivun rakenne (HTML), ulkoasu (CSS) ja selaimessa tapahtuvat toiminnallisuudet (JavaScript). (Laine 2015.)

2 Toteutuksen määrittely

Projektin alkuun suunnittelin, mitä teknologioita käytän ja kuinka hyödynnän niitä. Projektin etenemisen kannalta oleellisin valinta on sopivan JavaScript Frameworkin valitseminen. En ollut juurikaan käyttänyt uusimpia JavaScript Frameworkejä, joten edessäni oli paljon opettelua. Työssäni käytän päivittäin Knockout.js JavaScript Frameworkia, mutta se on hieman vanhentunut ja monimutkainen, joten ei näin ollen soveltunut tähän projektiin.

2.1 JavaScript Frameworkin valitseminen

JavaScript Framework tarkoittaa sovelluskehystä, jossa sovellus toteutetaan JavaScript-ohjelmointikielellä. Sovelluskehys antaa apuvälineitä ja suuntaviivoja, joilla sovelluskehittäjän on helpompi toteuttaa sovellusta. (Clifton 2003.) JavaScript Frameworkit ovat tarkoitettu web-ohjelmointiin eli verkkosivujen interaktiivisuuden tuottamiseen.

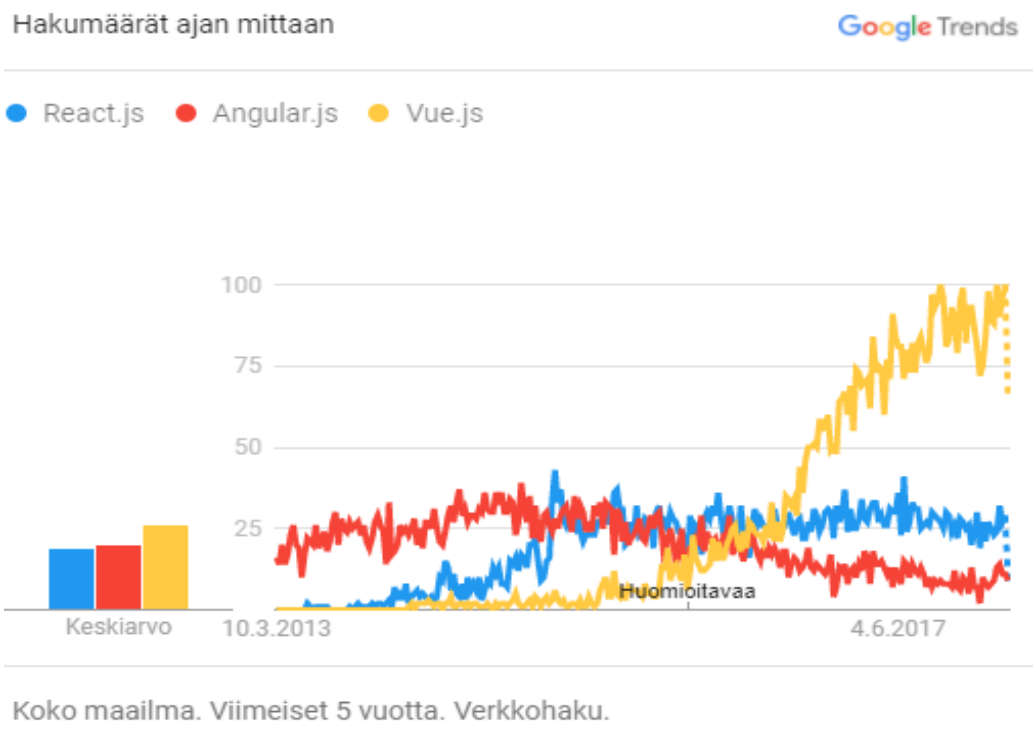
JavaScript Frameworkien ikä on tyypillisesti melko lyhyt ja ne kehittyvät ominaisuuksiltaan nopeasti. Jokin, mikä on ollut viisi vuotta sitten suosituin, saattaa olla nyt unohdettu ja korvattu uudemmilla teknologioilla. (Allen 2018.) Tämän takia uusien teknologioiden opiskelu on IT-alalla jatkuvaa, tärkeää ja merkittävänä syynä siihen, miksi toteutan verkkoportfolion itselleni uudella teknologialla.

2.2 Suosituimmat JavaScript Frameworkit

Tämän hetken kolme suosituinta JavaScript Frameworkia ovat React.js, Vue.js ja Angular.js. Näillä on toteutettu paljon isoja verkkopalveluita ja myös työllistävät merkittävästi kaikista käytetyistä Frameworkeista. (Elliott 2017.)

Selvitin Google Trends -työkalulla, mikä näistä kolmesta JavaScript Frameworkeista on haetuin. Google Trends -tulokset ovat vertailukelpoisia, sillä jokainen tietopiste on jaettu maantieteellisen sijainnin hakujen kokonaismäärällä ja sen edustamalla ajanjaksolla, jotta suhteellista suosiota voidaan verrata (Google Trends 2018).

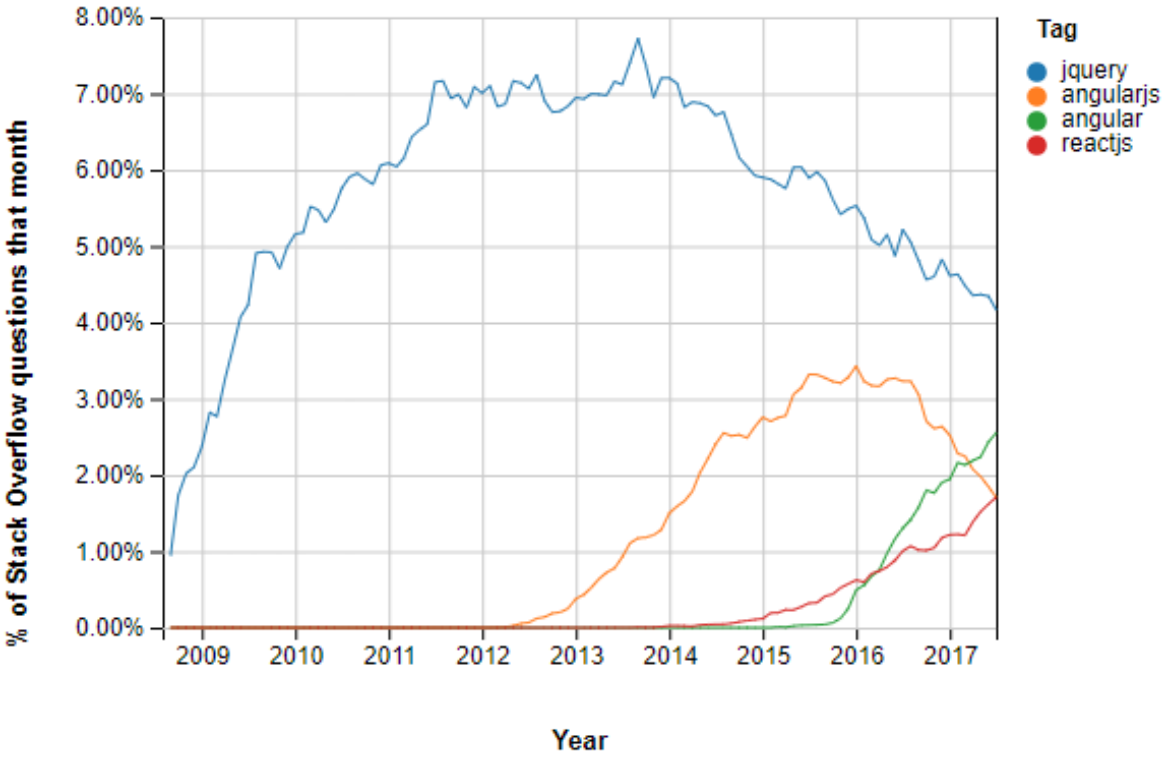
Kuviosta 1 voi päätellä, että viimeisen viiden vuoden aikana Vue.js:n hakumäärät ovat kasvaneet Angular.js:n ja React.js:n ohi. Hakuja on tapahtunut todennäköisesti eniten, kun etsitään apua kyseisellä Frameworkilla kehittäessä tai kun haetaan tietoa hakijalle tuntemattomasta Frameworkista. Tästä voi päätellä, että Vue.js:n suosio on kasvamassa.



Kuvio 1. Frameworkit Google Trendsissä

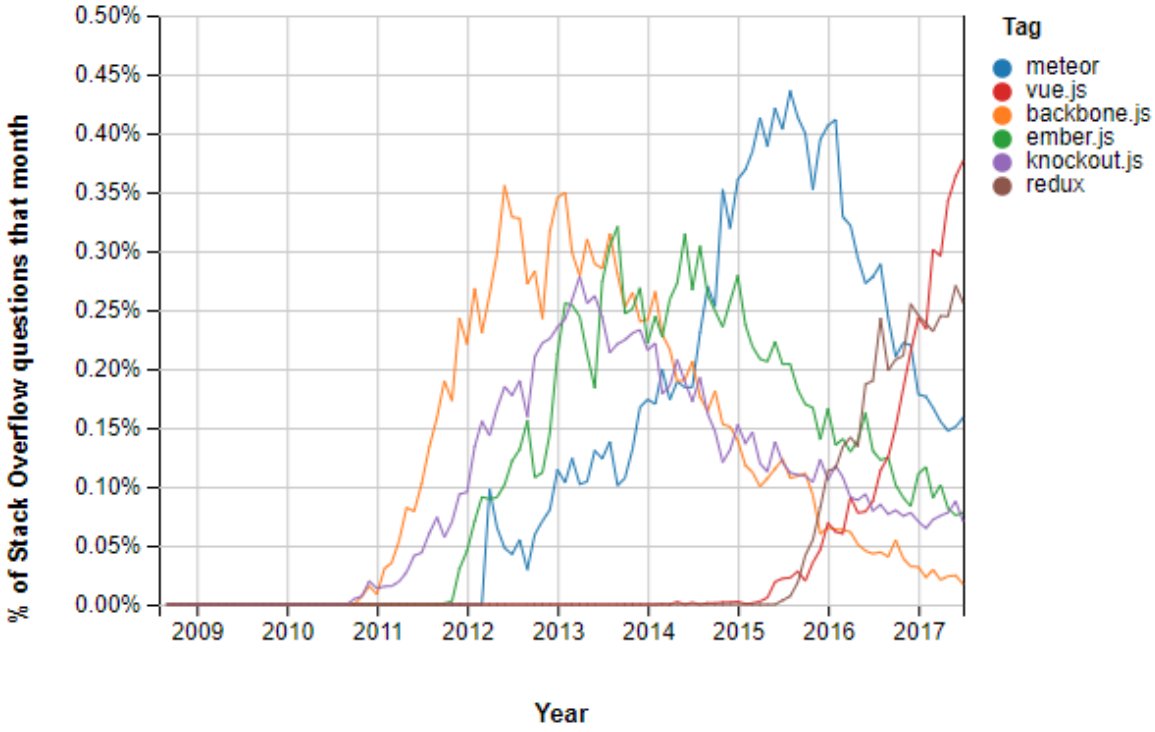
Tein vertailuja myös Stack Overflow -verkkosivun Stack Overflow Trends -työkalulla. Stack Overflow on maailman isoin ohjelmistokehittäjäyhteisö yli 50 miljoonalla kuukausittaisella käyttäjällä (Stack Overflow Insights 2018). Stack Overflow Trends -työkalulla voi verrata Stack Overflow:ssa tehtyjä hakuja vuodesta 2008 lähtien. Työkalu ehdotti suosituimmiksi hauiksi "JavaScript Frameworks" ja "Smaller JavaScript Frameworks".

Kuviosta 2 voi päätellä jQueryn olleen pitkään haetuin JavaScript Framework Stack Overflow:n sivuilla. Kuvasta näkee, että jQueryn suosio alkoi laskea Angularin ensimmäisen version julkaisun jälkeen. Samaa kaavaa seuraten vanhemman Angularin suosio alkoi laskea Angularin uusimman version ja Reactin myötä. Kuvan perusteella vuoden 2017 puolessa välissä Angularin uusin versio on ollut maailmanlaajuisesti Reactia suositumpi, mutta uskoisin maailmanlaajuisen tilanteen olevan tällä hetkellä varsin tasan. Tässä kuvaajassa ei ole Vue.js:ää, sillä se on vasta kasvattamassa suosiotaan ja yhteisö on vielä pieni verrattuna kuvan muihin JavaScript Frameworkeihin.



Kuvio 2. JavaScript Frameworks (Stack Overflow 2018a.)

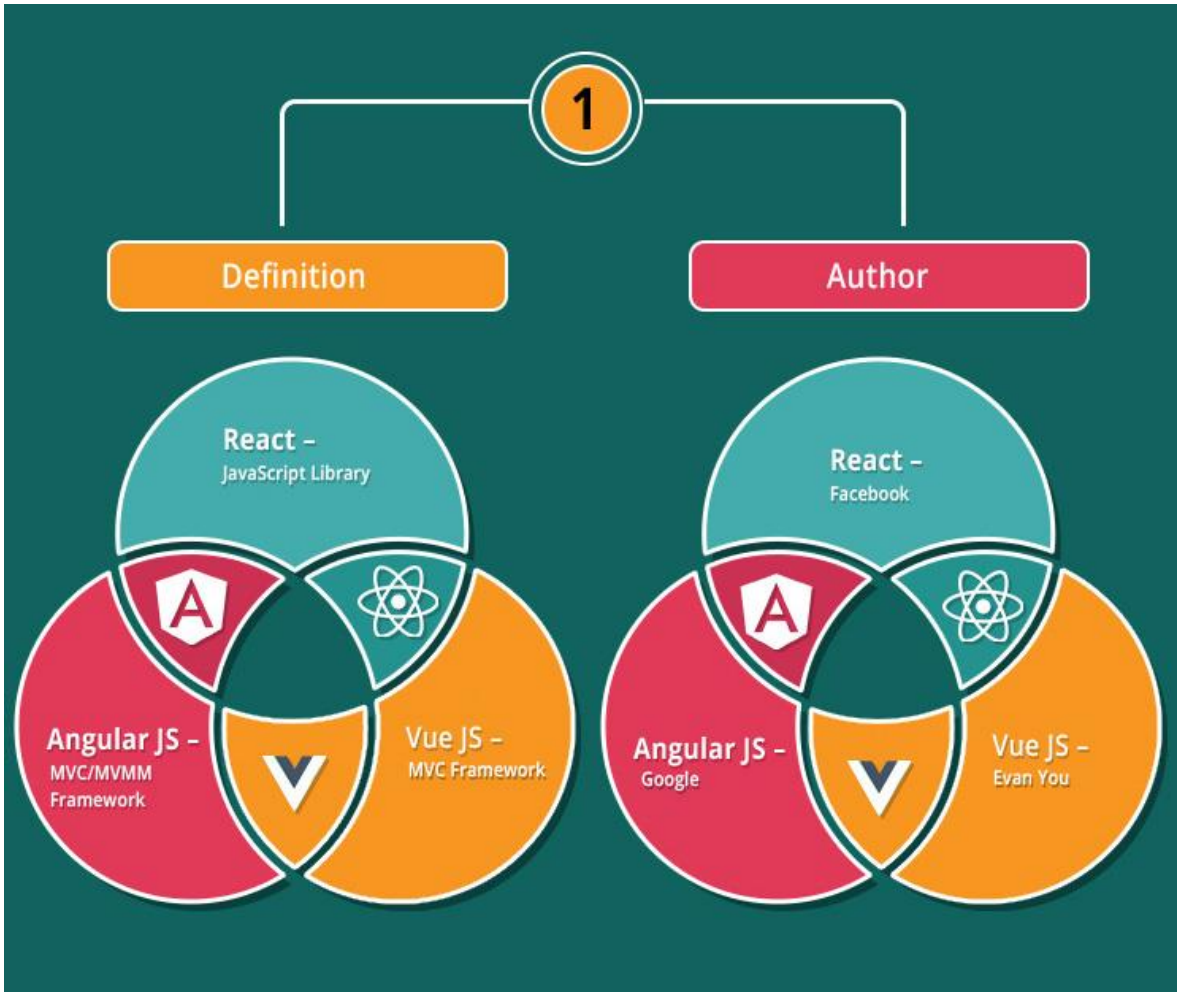
Kuviosta 3 voi päätellä Vue.js:n olleen Stack Overflow:n haetuin pienempi Framework vuoden 2017 puolivälissä. Nykyisellä kasvuvauhdilla Vue.js tulee todennäköisesti löytymään käytetyimpien Frameworkien joukosta.



Kuvio 3. Smaller JavaScript Frameworks (Stack Overflow 2018b.)

2.3 JavaScript Frameworkien vertailu

Kuviossa 4 havainnollistettuna kolmen JavaScript Frameworkin määritelmä ja tekijä. Seuraavissa kappaleissa käsittelen tarkemmin, miten arvioin näiden Frameworkien soveltuvuutta tähän projektiin.



Kuvio 4. Framework kuvaaja (Eduonix 2017.)

Angular.js on Googlen kehittämä ja ylläpitämä avoimen lähdekoodin JavaScript Framework. Angularin arkkitehtuuri on MVC eli Model-View-Controller, joka tarkoittaa mallia, näkymää ja käsittelijää. Malliin varastoidaan tieto, käsittelijä käsittelee tietoa ja näkymässä näytetään tieto (Microsoft 2018).

Lukemieni artikkelien (Cordle 2017; Aguinaga 2018.) perusteella Angular.js on jäykkä ratkaisu ja oppimiskäyrä on melko pitkä. Jäykällä tarkoitan, että tekijällä ei ole vapaita käsiä päättää, miten jokin ratkaisu toteutetaan, vaan Framework pakottaa tiettyyn malliin toteutuksessa. Tämä on hyvä ominaisuus, jos on isompi tiimi kehittämässä, jolloin koodi pysyy

mahdollisimman samanlaisena tekijästä riippumatta. Angular.js ei kuitenkaan palvele minun tarpeitani, vaan on turhan ylimitoitettu tähän projektiin.

React.js on Facebookin kehittämä ja ylläpitämä avoimen lähdekoodin JavaScript Framework. Kyseessä ei ole oikeastaan Framework vaan Library eli kirjasto, sillä React.js koostuu komponenteista, eikä pakota käyttäjäänsä tietynlaiseen malliin. Toisin kuin Angular.js on MVC arkkitehtuurin Framework, React.js käsittää vain MVC:n view-tason eli näkymän. (Reactjs 2018.)

Reactilla on tehty Facebookin lisäksi lukuisia suosittuja verkkosivuja kuten Airbnb, Dropbox, Imgur, Instagram, Netflix, Paypal, Periscope, Reddit, Uber, Tesla ja WhatsApp. (Medium 2018.) Mediumin artikkelin perusteella voi todeta, että valtaosa sosiaalisen median verkkosivuista on rakennettu Reactilla.

Vaikka React.js on luultavasti tämän hetken trendikkäin ja suosituin JavaScript Framework, siinäkin on ongelmansa. React.js antaa todella vapaat kädet toteutukseen ja oppimiskäyrä on Angularia pidempi. React.js on valtavan yrityksen tekemä Framework, tehty palvelemaan parhaiten alkuperäistä tarkoitustaan eli monimutkaisia ja laajoja käyttöliittymiä.

Vue.js on avoimen lähdekoodin JavaScript Framework, jonka kehitti aluilleen Evan You, entinen Googlen työntekijä ja Angularin kehittäjä. Evan You päätyi kehittämään Vue.js:n, sillä hän halusi Angularia kevyemmän Frameworkin ilman ylimääräisiä konsepteja. (Cromwell 2016.) Vue.js:n on siis tarkoitus olla helposti omaksuttava ja ketterä Framework. Se on kehitetty pienempään tarpeeseen, kuin suuren kompleksisen verkkosivun tekemiseen. Toisaalta Vue.js:llä onnistuu myös monimutkaisen kokonaisuuden tekeminen (Vuejs 2018.)

Vue.js erottui edukseen opittavuudellaan ja ketteryydellään. Vue.js palvelee selkeästi parhaiten tämänkaltaista yksinkertaisempaa projektia. Näinpä valitsin käyttäväni Vue.js frameworkia tähän projektiin.

2.4 SPA, MPA vai PWA

JavaScript Frameworkin valinnan jälkeen seuraava olennainen valinta oli päättää, millaisen toteutuksen teen. Vaihtoehtoja oli kolme. SPA eli Single-page application, MPA eli Multi-page application ja PWA eli Progressive Web Application.

Multi-page application tarkoittaa perinteistä monisivuista tapaa tehdä web-sovellus. Sivusto on jaettu eri näkymiin, jotka ladataan erikseen uuteen näkymään tai sivuun siirtyessä. Jokainen datan muutos lataa sivun uudestaan. Hyviä puolia ovat helppo hakukoneoptimointi ja soveltuvuus laajoihin sivukokonaisuuksiin. Perinteisesti isojen yritysten sivut tai on toteutettu MPA toteutuksena, sillä sisältöä ja palveluita on runsaasti, eikä niitä saa organisoitua järkevästi ainoastaan yhdelle sivulle. (Schneider, 2018; Skólski, 2018.)

Single-page application tarkoittaa yhden sivun sovellustoteutusta. Single-page application on sopiva valinta, kun sisällön saa organisoitua järkevästi mahtumaan yhteen ylhäältä alas rullattavaan sivuun. SPA on yleensä käyttäjäystävällisempi ja toimii mobiililla paremmin, kuin MPA, sillä SPA toteutustapa laittaa tekijän suunnittelemaan sivuston responsiiviseksi, helposti selattavaksi ja hyvin organisoiduksi. (Schneider, 2018; Skólski, 2018.)

SPA:t ovat yleistyneet älypuhelimien yleistyessä, sillä ne ovat helpompia käyttää älypuhelimella, kuin MPA:t. Kuviosta 5 näkyy, että SPA:n hakumäärät Googlessa ovat kasvaneet samoihin aikoihin kuin älypuhelimet ovat yleistyneet.

Kuvio 5, Single-page application

Single-page applicationissa hakukoneoptimointi on vaikeampaa, kuin MPA:ssa, sillä SPA:n data ladataan AJAX:lla päivittämättä varsinaista sivua. (Skólski 2018.) Hakukonebotit, crawlerit, eivät osaa kunnolla havaita SPA-sivun tietoja, joten SPA-sivut saavat

yleensä huonomman näkyvyyden hakukoneissa, verrattuna helposti hakukoneoptimoitaviin MPA-sivuihin. Tämän takia SPA-sivuissa pitää nähdä enemmän vaivaa tehdäkseen toteutuksen helpommin luettavaksi hakukoneboteille. (Google 2018.)

Progressive Web Application eli PWA on alkuaan Googlen vuonna 2015 esittelemä malli, jonka suosio on ollut kasvussa. PWA on sekoitus perinteistä web-sivua tai -sovellusta ja mobiilisovellusta ja pyrkii yhdistämään näiden parhaat puolet. (Pääkkö 2017.) PWA rakennetaan SPA toteutuksen päälle, joten toimiva SPA on PWA:n alkukriteeri. (Muskardin 2017.)

Jotta web-sovellus voidaan luokitella PWA:ksi, sen pitää täyttää Googlen (2018) mukaan kolme vaatimusta, jotka pohjautuvat hyvän käyttäjäkokemuksen kriteereistä. Vaatimuksia ovat luotettavuus, nopeus ja sitouttavuus sekä viehättävyys.

Ollakseen luotettava, sovelluksen pitää käynnistyä sulavasti riippumatta verkkoyhteyden tilasta. Sovelluksen ei tarvitse toimia täydellisesti yhteydettömässä tilassa, vaan käynnistymisen jälkeen näyttää jotain tilaa. Tämä toteutuu käyttäen Service Workeria, eli taustalla ajettavaa skriptiä. Nopeuden pitää ilmetä sovelluksen ja käyttäjän välisellä nopealla vuorovaikutuksella. Animaatioiden pitää olla pehmeitä ja rullaamisen sulavaa. Viehättävyydellä tarkoitetaan, että sovelluksen pitää vaikuttaa natiivilta mobiilisovellukselta. (Google 2018.)

Kun PWA:n ehdot täyttyvät, älypuhelimen selain ehdottaa sivua lisättäväksi sovellusluetteloon, jonne tulee sovelluksen käynnistyskuvake. Kuvakkeelta käynnistetty PWA-sovellus avautuu koko ruudulle ilman selaimelle ominaisia hakupalkkeja. Lisäksi on mahdollista määritellä push-notifikaatioita eli ilmoituksia, kun esimerkiksi sivustolle lisätään uutta sisältöä. Loistava esimerkki PWA-sovelluksesta on Twitter Lite: https://mobile.twitter.com/home.

2.5 Valitsemani sovellustyyli

Verkkosivuston sisältö on erinomaisesti mahdutettavissa yhdelle sivulle ja Vue.js on ilmeisesti loistava SPA-sovelluksissa. Lisäksi SPA-sovellus mahdollistaa WPA-toteutuksen, jos aikaa riittää. Mielestäni SPA on siis selkeä valinta, kun MPA:n ainoa selkeä etu on parempi hakukoneoptimoitavuus.

2.6 Käyttöliittymäkomponenttikirjasto

Käyttöliittymäkomponenttikirjasto tai lyhyemmin UI-komponenttikirjasto (User Interface) nopeuttaa sovelluskehitystyötä, kun sovelluskehittäjän ei tarvitse käyttää aikaa käyttöliittymän toteuttamiseen alusta alkaen. UI-komponenttikirjaston merkitys ajan ja resurssien säästössä korostuu varsinkin ison yrityksen sovelluskehityksessä, kun ei tarvitse käyttää aikaa samojen asioiden toteuttamiseen toistamiseen. (Scott & Mondago 2018.) UI-komponenttikirjastoa käyttäessä toteutuu myös paremmin ohjelmistokehitysperiaate, DRY (Don't Repeat Yourself), joka kehottaa välttämään samojen asioiden toistamista (Peters 2012).

Selvitin mitä UI-komponenttikirjastoja Vue.js:lle on saatavilla. Jonathan Saring (2017) vertaili artikkelissaan yhtätoista UI-komponenttikirjastoa Vue.js:lle, joista hän suositteli Vuetify.js -kirjastoa parhaana. Vuetify.js:n tyyli pohjautuu Googlen kehittämään Material Designiin ja on yksi suosituimmista UI-komponenttikirjastoista Vue.js:lle noin 9.5 tuhannella GitHub tähdellä. Mitä suurempi ja suositumpi JavaScript- tai komponenttikirjasto on, sitä varmemmin se toimii. Tämä perustuu siihen, että iso kehittäjäyhteisö löytää bugit ja korjaa ne nopeasti.

2.7 Sisällönhallintajärjestelmä

Sisällönhallintajärjestelmä (CMS, content management system) on järjestelmä, jolla voi luoda, hallita ja julkaista sisältöä jossakin järjestelmässä, kuten esimerkiksi yrityksen Intranetissä tai yksityishenkilön blogissa. Sisällönhallintajärjestelmä on suosittu tapa julkaista sisältöä verkkosivuilla ja useat palveluntarjoajat tarjoavat myös verkkosivujen isännöinnin, sekä valmiita verkkosivupohjia. Edellä mainittua, erityisesti verkkosivuihin suuntautunutta ratkaisua kutsutaan julkaisujärjestelmäksi (WCMS, web content management system). (Kohan 2010.)

Sisällönhallintajärjestelmät ovat suosittuja ja käteviä, sillä käyttäjän ei tarvitse osata yhtä paljoa teknisiä asioita kuin perinteisellä verkkosivujen julkaisumenetelmällä, joka on toteuttaa sisältö lähdekoodiin ja julkaista se palvelimella (Kohan 2010). Suosituin sisällönhallintajärjestelmä on Wordpress, joka on avoimen lähdekoodin CMS. WordPressin mukaan 30% verkkosivuista käyttää WordPressiä ja yli 60 miljoonaa ihmistä pyörittää verkkosivujaan WordPressillä. (Wordpress 2018.)

Sisällönhallintajärjestelmän käyttöönotto toimeksiantajan verkkosivuille on tärkeä prioriteetti, jotta toimeksiantaja pystyisi helposti päivittämään verkkosivujensa sisältöä. Sisällönhallintajärjestelmä ei kuitenkaan ole minimivaatimus ja olen lupautunut ylläpitämään sisältöä, kunnes saan sisällönhallintajärjestelmän toimimaan.

Sisällönhallintajärjestelmää ei kannata toteuttaa itse, sillä valmiita ratkaisuja on lukuisia ja ne ovat todennäköisesti ison joukon käyttämiä, toimivaksi tehtyjä ja turvallisempia kuin omat tekeleet. Varteenotettavaksi CMS-ratkaisuksi Vue.js:lle löysin ButterCMS:n ja Storyblokin, jotka ovat molemmat API-pohjaisia ratkaisuja.

2.8 Sovelluksen virtuaalinen paketointi

Sovelluksen virtuaalinen paketointi eli konttien tekeminen on kasvattanut suosiotaan ja liittyy vahvasti sovelluksen julkaisuun. Tämän hetken suosituin konttisovellus on Docker, jota kuvataan ohjelmistoalalla tällä hetkellä suorastaan kuumaksi (Vaughan-Nichols 2018). Julkaisun suunnitteluvaiheessa päädyin käyttämään Dockeria, sillä sen tarjoamat edut ovat selkeät ja Dockerin osaaminen on hyödyksi työmarkkinoilla.

> Docker on avoimen alustan sovellus, joka pystyy pakkaamaan sovelluksen kaikkien tarvittavien riippuvuuksien kanssa yhteen virtuaaliseen pakettiin, eli konttiin. Tätä teknologiaa kutsutaan konttiteknologiaksi. Näitä kontteja ohjelmistokehittäjät ja järjestelmäasiantuntijat voivat rakentaa, lähettää ja ajaa helposti hajautettujen järjestelmien sisällä. Teknologia tuo valtavia etuja yrityksille uudenlaisen siirrettävyyden, skaalautuvuuden, nopeuden, jakamisen ja ylläpidon muodossa. Yksinkertaistettuna Dockerin kontit ovat ajossa olevia Docker-kuvia (engl. image). Kuvat ovat valmiiksi rakennettuja ja helposti siirrettäviä, stabiileita perusyksiköitä. Yhdestä kuvasta voi olla useampi kontti ja kyseiset kontit koostuvat yhdestä peruskuvasta. Konttien rakentaminen on yksinkertaistettu skriptipohjaisella Dockerfiles'lla, jonka avulla rakennusprosessi voidaan automatisoida. (Vase 25.2.2016).

3 Toteutus

Toteutus osiossa kerrotaan toteutuksen ja julkaisun etenemisestä suurin piirtein siinä järjestyksessä kuin toteutukseni eteni. Kuvailen käyttämiäni ratkaisuja, esittelen toteuttamiani näkymiä ja kerron yleisesti työn kulusta.

3.1 Verkkotunnus

Projektin alussa varasin toimeksiantajalleni domainin eli verkkotunnuksen. Verkkotunnus on verkkosivun nimi ja osoite, jolla Internetin käyttäjät pääsevät verkkosivuille. Domain Name System (DNS) eli nimipalvelujärjestelmä kääntää verkkotunnukset IP-osoitteiksi. Verkkotunnus piilottaa verkkosivun IP-osoitteen ja näyttää käyttäjälle helpommin muistettavan verkkotunnuksen, kuten esimerkki.com. (Gil 2018.) Domain Name System eli DNS on Internetin nimipalvelujärjestelmä, joka muuntaa verkkotunnuksia IP-osoitteiksi.

Toimeksiantajani halusi käyttää omaa nimeään verkkotunnuksena sekä halusi verkkotunnuksen päätteeksi kaupalliseen tarkoitukseen käytetyn .com -päätteen. Verkkotunnukseksi valikoitui kallepitkanen.com.

Oma nimi verkkotunnuksena on hyvä tapa markkinoida itseään, sillä se on helppo muistaa ja parantaa hakukoneessa löytymistä. Verkkotunnuksessa kannattaa käyttää ainoastaan kansallisia symboleita, jotta verkkotunnus olisi mahdollisimman helppo kirjoittaa Suomenkin ulkopuolella. Tämän takia nimen ääkkönen on vaihdettu aakkoseksi.

Verkkotunnuksen ylätasoksi .com on hyvä ratkaisu, sillä se viittaa kaupalliseen tarkoitukseen (commercial). Taitelijaportfolion on tarkoitus markkinoida henkilöä, joten tarkoitus on kaupallinen.

3.2 Verkkotunnuksen hankinta

Vertailin Internet-artikkeleilla erilaisia verkkotunnusvälittäjiä saadakseni hyvän käsityksen, mistä kannattaisi varata verkkotunnus. Ensimmäisenä minulle tuli mieleeni aiemmin käyttämäni Gandi.net, sillä sieltä olen hankkinut itselleni verkkotunnuksen. Halusin kuitenkin saada laajemman yleiskuvan mikä olisi tällä hetkellä paras verkkotunnusvälittäjä, sillä Gandin hallintanäkymä oli omasta kokemuksestani vanhentunut.

Luin verkkoartikkelit https://makeawebsitehub.com/reviews/domain-registrars/ ja https://bloggingthing.com/best-domain-name-registrars-2017-2018, joista molemmat suosittelivat parhaana vaihtoehtona Namecheap:ia. Tärkeimmät perustelut olivat käyttäjäystävällinen käyttöliittymä, järkevät hinnat ja nopea sekä luotettava tukipalvelu.

Verkkotunnus kallepitkanen.com maksoi 8,95 € + 0,15 € ICANN-maksu (rekisteröintimaksu) = 9,09€. Valitsin käyttööni ilmaisen WhoIsGuard-palvelun, joka suojaa yksityistiedot WhoIs -pyynnöiltä, eli verkkotunnuksen omistajan henkilötietokyselyiltä. (NameCheap 2018.)

3.3 Kehitysympäristö

Verkkosivujen toteuttamisen aloitin asettamalla kehitysympäristöni kuntoon. Ohjelmistokehityksessä suositaan Unixia tai Unix-tapaisia käyttöliittymiä, kuten MacOS tai Ubuntu, sillä niissä on hyvin toimiva komentorivi ja saatavilla runsaasti vapaan lähdekoodin ohjelmistoja. Tietokoneelleni oli asennettuna Xubuntu-käyttöjärjestelmä, joka on kevennetty versio Ubuntu-käyttöjärjestelmästä ja soveltuu mainiosti ohjelmistokehitykseen. Xubuntua en erikseen valikoinut tätä projektia varten, vaan sen olin asentanut aiemmin. Koska Xubuntu oli itselleni tuttu käyttöjärjestelmä, en kokenut tarpeelliseksi tutkia muita vaihtoehtoja käyttöliittymäksi.

3.4 Versionhallinta

Projektia varten loin tietokoneelleni thesis-kansion, josta tein GitHub repositoryn eli säilytyspaikan GitHub-versionhallintaa varten. Git-versionhallinta on järjestelmä, johon tallennetaan tiedostoon tai projektiin tehdyt muutokset. Versionhallinnassa voi vertailla tehtyjä muutoksia ja tarkistaa, kuka teki muutokset. Tallennettu muutos on uusi versio ja versionhallinnassa voi liikkua tehtyjen versioiden välillä edes takaisin. Versionhallinta myös mahdollistaa projektin palauttamisen, jos projekti jostain syystä katoaa tietokoneelta tai projektiin ei enää pääse käsiksi. (Git 2018.) Edellä mainituista syistä versionhallinta on mielletty pakolliseksi osaksi sovelluskehitystä, joten sen käyttäminen tässäkin projektissa on selvä asia.

3.5 Käyttöliittymäsuunnitelma

Tein käyttöliittymäsuunnitelmat Mockplus -työkalun kokeiluversiolla. Suunnittelemani käyttöliittymä käsittää neljä näkymää: Main page eli etusivu, Biography eli artistikuvaus, Gal-

lery eli kuvagalleria ja Contact eli yhteystiedot. Hyväksytin käyttöliittymäsuunnitelmat toimeksiantajallani ja teimme pieniä muutoksia suunnitelmiin. Tarkemmat Mockplus-työkalulla tehdyt käyttöliittymäkuvaukset löytyvät osiosta liitteet.

3.6 Toteutuksen aloittaminen

Aloitin toteutuksen asentamalla Vue.js:n komennolla: ”npm install -g vue-cli”. Tämän jälkeen asensin vuetify-pwa projektipohjan komennolla: ”vue init vuetifyjs/pwa thesis”. Ohjeet asentamiseen ja projektin aloittamiseen löytyivät Vuetify:n GitHub repositorylta https://github.com/vuetifyjs/pwa. Projektipohja käynnistyi onnistuneesti selaimessa komennolla ”npm run dev”. Tässä vaiheessa käytin noin kaksi päivää projektipohjaan tutustumiseen ja oppaiden lukemiseen.

3.7 Verkkosivujen taustakuva

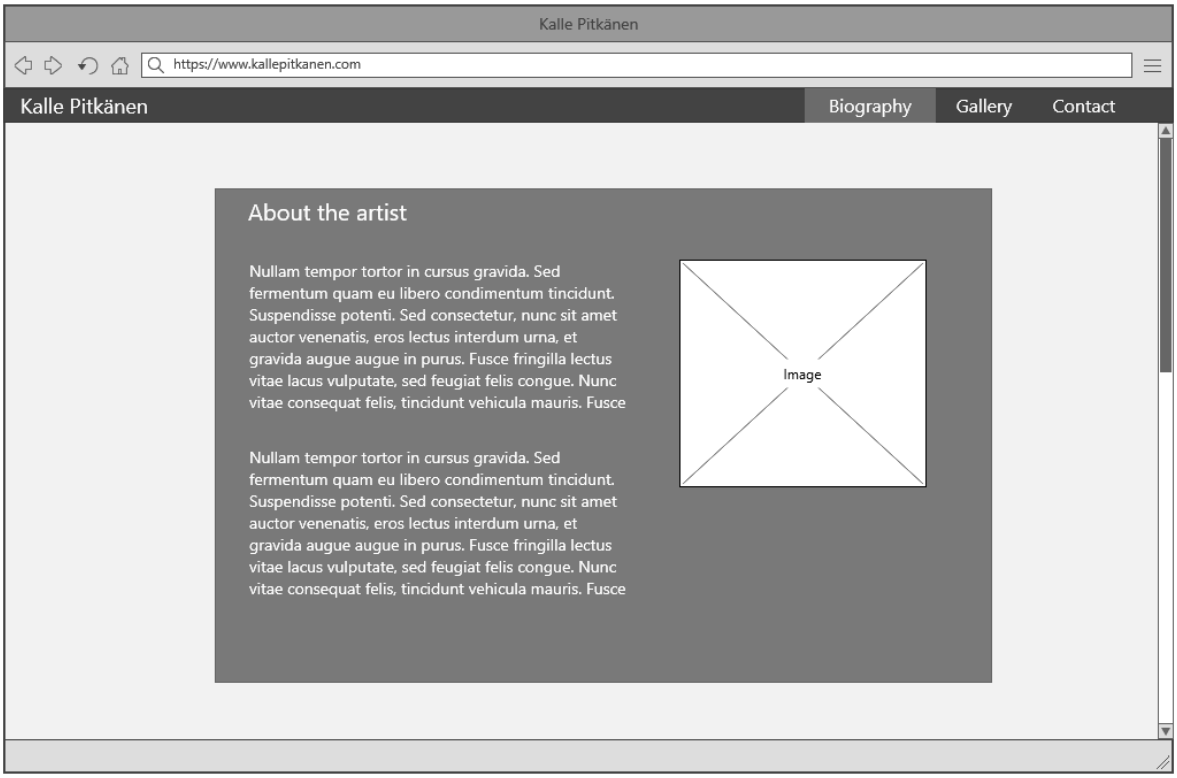
Mielestäni verkkosivuprojektin koodauspuoli on sujuvaa aloittaa taustakuvan ja layoutin eli pohjarakenteen toteuttamisella. Kun saa taustan ja layoutin responsiiviseksi, on oppinut paljon oleellista käyttämänsä frameworkin toiminnasta. Responsiivisen layoutin pohjalle on hyvä alkaa toteuttamaan sisältöä.

Verkkosivu tai verkkosivun elementit sisältävät aina taustakuvan tai taustakuvavärin, joka määritellään yleensä CSS-säännöillä. Taustaväri on helppo ja turvallinen valinta, sillä taustaväri skaalautuu hyvin erikokoisille näytöille eli on responsiivinen. Taustakuvan kanssa joutuu näkemään enemmän vaivaa, sillä taustakuvan pitää mukautua erikokoisille ruuduille. Esimerkiksi pelkästään 1920x1080 pikselin resoluutiolle suunniteltu taustakuva näyttää huomattavan erilaiselta isomman resoluution tai pienemmän resoluution näyttölaitteella. Erityisesti pienemmät resoluutiot ovat tärkeitä ottaa huomioon, sillä verkkosivuja käytetään nykyään paljon mobiililaitteilla.

Taustakuvalla verkkosivujen ilmeestä saa persoonallisen ja hyvin toteutettuna näyttävän. Hyvin toimivalla ja näyttävällä taustakuvalla saadaan kiinnitettyä vierailijan huomio, jolloin hän todennäköisesti viettää pidemmän ajan verkkosivuilla. Lisäksi näyttävät verkkosivut jäävät paremmin mieleen. Taustakuvan olisi hyvä liittyä verkkosivujen aiheeseen ja taustakuva voi jo itsessään kertoa, mitä verkkosivut käsittelevät. (Karol K 2.6.2016.)

Suunnittelin käyttöliittymäsuunnitelmissani, että taustakuva on koko ruudun kokoinen ja se pysyy paikoillaan. Ruudun kokoa pienennettäessä, taustakuva kutistuu keskitetysti eli keskikohta pysyy koko ajan keskellä ja kuvan reunojen kadotessa. Taustakuvan päälle tulee sisältö omiin ”laatikoihinsa” eli näkymiin, jotka olen nimennyt sisältönsä mukaan; Home

page, Biography, Gallery ja Contact. Kuvassa 1 on esimerkkinä Biography -näkymän käyttöliittymäsuunnitelma.



Kuva 1. Biography -näkymä

Tavoitteena on persoonallinen ilme, koska verkkosivut toteutetaan taiteilijalle. Suunnittelimme toimeksiantajan kanssa, että hän voisi maalata taustakuvan. Koska taustakuvan pitää olla responsiivinen, paras lopputulos on kuvalla, jossa ei ole selkeää kuviota.

Sain toimeksiantajalta muutaman kuvavaihtoehdon, jotka olivat .jpg-formaatissa. Kuvat olivat 4K-resoluutioisia ja kooltaan noin 5-10 megatavua kappale. Alkuperäisiä kokoja käytettäessä verkkosivujen latausnopeus olisi laskenut huomattavasti. Pienensin kuvakokoa ImageMagick-ohjelmalla noin 150 kilotavuun. Säilytin resoluution korkeana, jotta kuva skaalautuisi hyvin myös 2K- tai 4K-näytöille. Gimp-ohjelmalla rajasin kuvasuhteen pystykuvasta vaakakuvaksi, jotta kuva skaalautuisi paremmin. Kuvasuhteeksi asetin 16:9, joka on näyttölaitteen normaali kuvasuhde.

Keksittyäni, mihin päin Vue-projektissa oli tarkoitus laittaa taustakuva, taustakuvan toteuttaminen oli yksinkertaista, eikä aiheuttanut hirveästi päänvaivaa.

3.8 Verkkosivujen layout

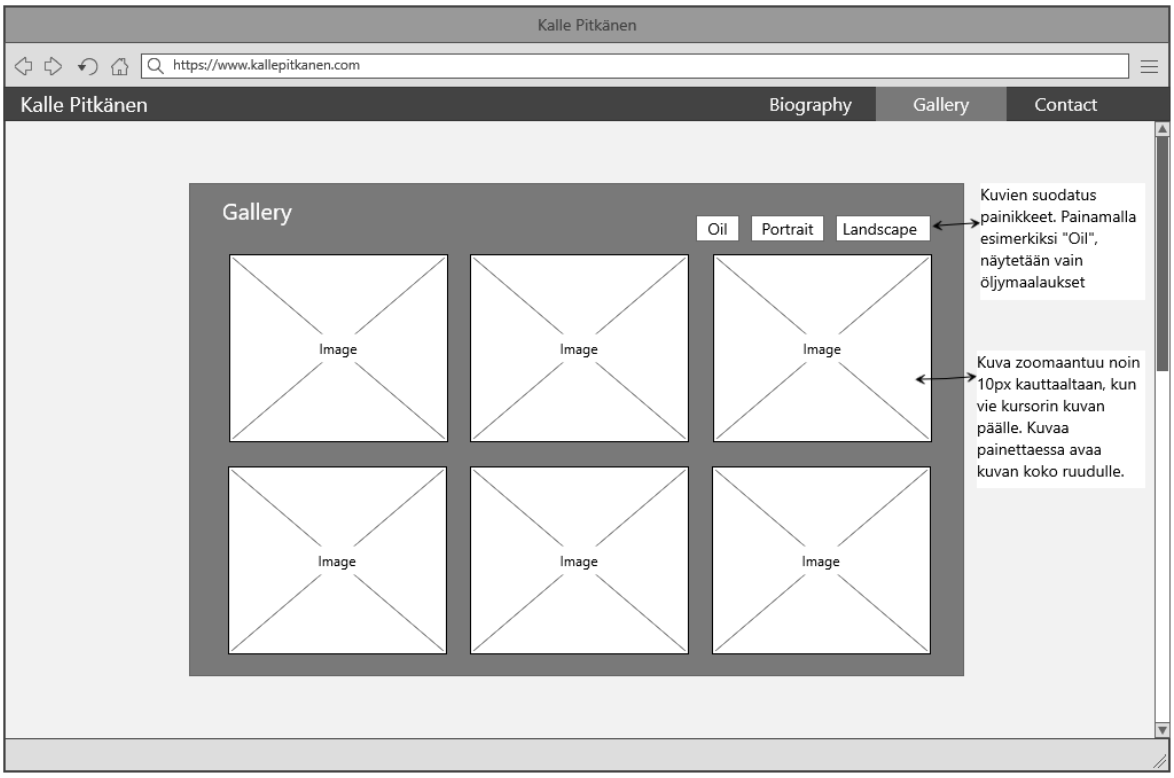
Taustakuvan jälkeen aloin toteuttamaan verkkosivujen layoutia eli pohjarakennetta. Vuetify-projektissa layoutin voi toteuttaa joko perinteiseen tapaan asettamalla HTML-luokkiin CSS:n flexbox -sääntöjä (W3Schools 2018) tai Vuetifyn omalla grid-systeemillä, joka sekin käyttää flexboxia. (Vuetify 2018.) Tavoitteenani oli oppia mahdollisimman paljon Vuetifysta, joten aloin toteuttamaan layoutia Vuetifyn grid-systeemillä.

Sain toteutettua layoutista hyvännäköisen version muutamassa päivässä, mutta en ollut tyytyväinen responsiivisuuteen. Vuetifyn dokumentaatiossa ei ollut mielestäni tarpeeksi selkeästi kuvattuna, miten mitäkin gridin propsia kuuluisi käyttää, enkä löytänyt Googlesta hakemalla yhtäkään esimerkkiprojektia, jossa olisi samankaltainen pohjarakenne. Propsit (properties) ovat Frameworkin omia parametrejä helpottamaan kehitystyötä.

Lopulta pääsin oikeaan käsitykseen, miten pohjarakenne kuuluisi toteuttaa. Jouduin tekemään siihen astisen toteutuksen lähes kokonaan alusta, joten opettelutyöhön meni useampi päivä työtä hukkaan. Vaikka Vuetifyn grid-systeemi muistutti erittäin läheisesti toisessa projektissa käyttämäni Bootstrap UI-kirjaston grid-systeemiä, Vuetifyn gridin käyttöönotto ei ollut yhtä helppoa. Ongelmia aiheutti, kun yritin käyttää propseja, jotka olivat lähes saman nimisiä Bootstrapissa ja Vuetifyssa, mutta toimivat hieman eri tavalla. Lisäksi yritin käyttää propseja samankaltaisesti kuin toisessa projektissani, vaikka siinä oli erilainen pohjarakenne.

3.9 Sisältö

Pohjarakenteen jälkeen seuraava askel oli toteuttaa varsinainen sisältö sisältölaatikoiden eli containerien sisälle. Edellisessä kohdassa olin saanut containerit responsiivisiksi. Kuvassa 2 container on tummanharmaalla värjätty alue keskellä layoutia. Sisällöllä tarkoitetaan kaikkea containerin sisällä olevaa, kuten kuvat ja tekstit.



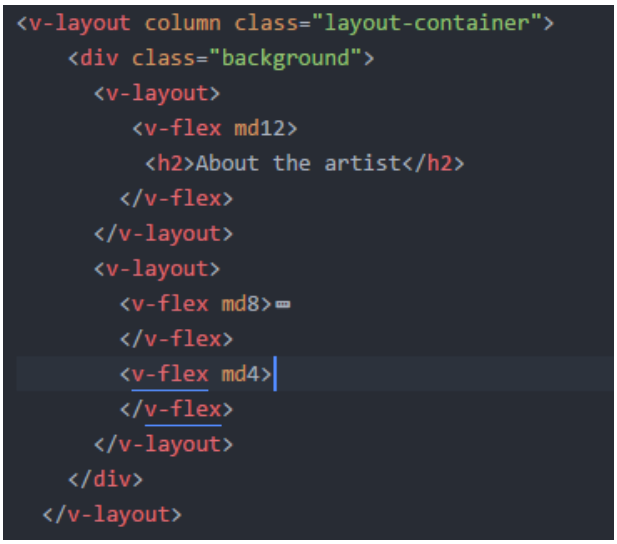
Kuva 2. Gallery -näkymä

3.10 Biography

Biography container sisältää ytimekkään tekstin toimeksiantajasta ja hänen kuvansa. Aloitin tästä containerista, sillä se oli etunäkymän jälkeen kaikista yksinkertaisin toteuttaa. Tekstit ja kuvan sain toimeksiantajalta.

Kuvasta 1 voi havaita containerin jakautuvan keskeltä kahteen palstaan. Kuvassa 3 nämä palstat ovat v-flex md8 ja v-flex md4. Kuvan v-layout on Vuetifyn grid-systeemin elementti, jonka on tarkoitus muodostaa omia pienempiä layout kokonaisuuksiaan. V-layout sisälle laitetaan v-flex elementit, jotka jakavat v-layoutin pienempiin soluihin. Grid-systeemi jakautuu kahteentoista soluun, joten täysi leveä alue merkitään 12 solua leveäksi. Täysi leveän alueen voi jakaa pienempiin alueisiin, kuten tässä tapauksessa vasen puoli on 8/12 leveä ja oikea puoli 4/12 leveä.

Oikeanpuolimmaiseen soluun tulevaa kuvaa joudun käsittelemään sen verran, että kutistin kuvakokoa, jotta sivut latautuisivat nopeasti.



Kuva 3. Biography -näkymän container

3.11 Contact

Contact container sisältää toimeksiantajan yhteystiedot sekä yhteydenottolomakkeen. Yhteystietoja havainnollistin yhteystietoa kuvaavalla ikonilla. Esimerkiksi puhelinnumeroa kuvaamaan käytin Font-Awesomen ikonia mobile-alt.

Yhteystietolomakkeen tekstikentät toteutin Vuetifyn v-text-field -komponentilla ja tekstikenttien validointiin löytyi hyvät esimerkit komponentin dokumentaatiosta. Validointi tarkoittaa, että käyttäjän syöttämä sähköpostiosoite pitää olla tietynlainen. Esimerkiksi sähköpostikentän syötteen pitää sisältää tietty määrä merkkejä ennen @-merkkiä, jonka jälkeen taas tietty määrä sallittuja merkkejä ja lopuksi esimerkiksi pääte .com.

Yhteystietolomakkeen alle sijoitin painikkeen kenttien tietojen tyhjentämiseen sekä tietojen lähettämiseen. Tietojen lähetys tapahtuu JSON-muotoisena API-kutsuna palvelimelle, joka lähettää tiedot eteenpäin sähköpostiosoitteeseen. JSON on yksinkertainen formaatti tiedon välittämiseen (Json.org 2018).

3.12 Gallery

Kuvagalleriaa varten etsin valmista JavaScript-kirjastoa, sillä oman toteuttamisessa olisi kulunut paljon aikaa ja valmiita ratkaisuja löytyy runsaasti. Parhaana ratkaisuna löysin PhotoSwipe.js:n, mutta sen käyttöönotto Vuelle olisi ollut työlästä. Lisää etsiessäni löysin Vue-Picture-Swipen, joka oli toteutettu PhotoSwipen pohjalta Vuelle. Vue-Pictur-Swipen käyttöönotto kävi kätevästi laittamalla containerin sisälle koodipätkän:

” `<vue-picture-swipe :items="items"></vue-picture-swipe>` ”.

Tiedoston Script-puolella items Arrayn sisälle laitettiin kuvien lähde, joten sain kuvat nopeasti paikoilleen. Jonkin verran jouduin muokkaamaan tyylejä ennen kuin sain käyttöliittymässä näkyvät suurentamattomat kuvat aseteltua siististi containerinsa sisälle. Tässäkin tärkeää oli säilyttää kuvasuhde oikeana, vaikka hieman kuvasuhdetta venyttämällä olisi saanut täytettyä symmetrisesti koko kuville varatun alueen.

Vue-Picture-Swipestä puuttui yksi oleellinen ominaisuus eli kenttä, johon saisi kuvan tiedot näkyville. Esitin GitHubissa Vue-Picture-Swipen tekijälle parannusehdotuksen kentästä ja paremmasta dokumentaatiosta. Kävimme keskustelua asiasta ja selvisi kyseisen ominaisuuden olevan tulossa lähiaikoina. Jään odottelemaan kauanko seuraavan Vue-Picture-Swipe version julkaisussa kestää ja jos siitä ei julkaista kuukauteen, niin harkitsen JS-kirjaston vaihtamista tai kuvien tietojen toteuttamista itse.

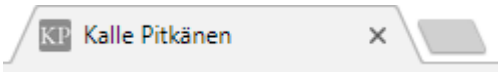
Suunnittelin etukäteen, miten saisin toteutettua sisällönhallintapalvelun kuvagalleriaan ja aloin lähes toteuttamaan suunnitelmaani. Toteutan todennäköisesti sisällönhallinnan Storyblok-palvelulla opinnäytetyöprojektin jälkeen. Projektin aikataulu ei antanut myöten Storyblokin käyttöönottoon tässä vaiheessa. Storyblok asennetaan toteutukseen ja koodiin määritellään, mitkä tekstit tai osiot vastaanottavat Storyblokilta sisältöä JSON-muotoisilla API-kutsuilla.

Käyttäjälle on graafinen käyttöliittymä, josta hän pääsee syöttämään sisältöä Storyblok-palveluun määritellyille kentille. Eli ratkaisun pitäisi olla helppokäyttöinen toimeksiantajalle, eikä vaadi teknistä osaamista sisällön päivittämiseen. Palvelun hinta oli 14 euroa kuukaudessa kaupalliseen käyttöön ja kehityskäyttöön ilmainen (Storyblok 2018).

3.13 Viimeistelyä

Kaikkien näkymien valmistuttua oli vuorossa vielä viimeistelyä. Asetin headerin eli yläpalkin navigaation painikkeet johtamaan oikeisiin näkymiin vue-routerin syntaksilla:
“ `<v-btn :to="{ name: 'biography'}" flat>Biography</v-btn>` ”.
Asetin mobiilinavigaation tulemaan esille ruutukoon pienennyttyä tietyn pisteen ali ja katoamaan tämän pisteen ylittyessä.

Suunnittelin ja tein favicon-ikonin, joka on selaimen palkissa näkyvä 16x16, 32x32 tai 64x64 pikselin kokoinen kuva. Alla olevassa kuvassa 4. esimerkki, miltä favicon-ikoni näyttää Google Chromella. Tein samalla useamman eri koon versiot kuvasta ennakoidakseni, jos saan tulevaisuudessa toteutettua PWA-toteutuksen, jolloin kyseisen kuvan saa tallennettua sovelluskuvaksi mobiililaitteelle.



Kuva 4. Favicon-icon

Toteutin footerin eli alapalkin Contact-näkymään. Monesti tällaisissa toteutuksissa footer olisi koko ajan staattisesti ruudun alareunassa, mutta mielestäni se olisi peittänyt turhaan taustakuvaa. Tein footerista pelkistetyn, sisältäen linkit toimeksiantajan Instagram- ja Facebook-tileihin. Tarvittavat yhteystiedot ovat esillä footerin yläpuolella contact-näkymän containerissa, joten yhteystietoja ei ollut tarpeen lisätä toistamiseen footeriin. Alla kuvassa 5. toteutettu footer.

Kuva 5. Footer

Tein nopeita ja helposti toteutettavia toimenpiteitä hakukonenäkyvyyden parantamiseksi, kuten lisäsin index.html:n eli oletustiedoston <head> -osioon otsikon ja kuvaukset.

3.14 Julkaisun suunnittelu

Ennen julkaisua perehdyin erilaisiin julkaisuvaihtoehtoihin. Varteenotettavana vaihtoehtona pidin Googlen Firebase Hosting -palvelua, joka olisi luultavasti ollut helpoin tapa julkaista verkkosivut. Firebase Hostingin ohjeiden mukaan olisi karkeasti ottaen pitänyt ladata ja asentaa palvelu muutamalla komennolla, konfiguroida firebase.json-tiedosto viittamaan dist-kansioon ja julkaista sovellus komennolla ”firebase deploy”. Lisäksi Firebase Hostingilla verkkosivut saavat automaattisesti SSL-sertifikaatin eli verkkosivulla on käytössä HTTPS-protokolla, joka takaa verkkosivuston turvallisen tiedonkulun.

Firebase Hosting vaikutti jopa liiankin helpolta ratkaisulta, kun tarkoituksena oli oppia uutta. Näinpä painoin Firebase Hostingin mieleeni hyvänä julkaisuratkaisuna, jos pitää saada nopeasti ja vaivattomasti julkaistua jotain. Docker on ollut kuuma sana jo pidemmän aikaa sovellusten julkaisuissa (Vaughan-Nichols 2018). Näinpä halusin kokeilla sitä tähän toteutukseen.

3.15 Julkaisu Dockerilla ja Dokkulla

Asensin Dockerin ja tein toteutukseeni Dockerfilen, joka on Dockerin konfiguraatiotiedosto. Dockerfile kertoo mitä Docker tekee ja näin automatisoi vaiheita. Sovelluksen rakennusvaiheessa minimoidaan automaattisesti kaikki tiedostot, jolloin ne vievät huomattavasti vähemmän tilaa. Konfiguroin sovellukseni pyörimään Docker containerissa Nginx-

palvelimella. Lopuksi Docker muodostaa Docker imagen, joka on ympäristöstä riippumaton ja kevyt kopio sovelluksesta. Docker imagen voi laittaa mille tahansa palvelimelle pyörimään. Testatakseni muodostuneen Docker imagen toimimisen, voin käynnistää sen virtuaalipalvelimen portilla 80 komennolla: ”docker run -p 8080:80 c014495fb2a0”. Komennon lopussa oleva ”c014495fb2a0” on imagen tunnus. Kyseisen komennon jälkeen saan sovelluksen auki selaimelleni haulla ”localhost:8080”.

Toimivan Docker imagen jälkeen piti löytää palvelu, johon asentaa Docker image, jotta se näkyisi julkisesti verkossa. Tähän tarkoitukseen löysin DigitalOceanin Dokku-dropletin. DigitalOcean on suuri pilvialustojen tarjoaja. Dokku on avoimen lähdekoodin PaaS (Platform as a Service) eli palvelualustan tarjoaja. Dokku vaikutti minut lupauksella helposta sovelluksen julkaisemisesta. Koko julkaisuketjun voi toteuttaa yhdellä versionhallinnan komennolla.

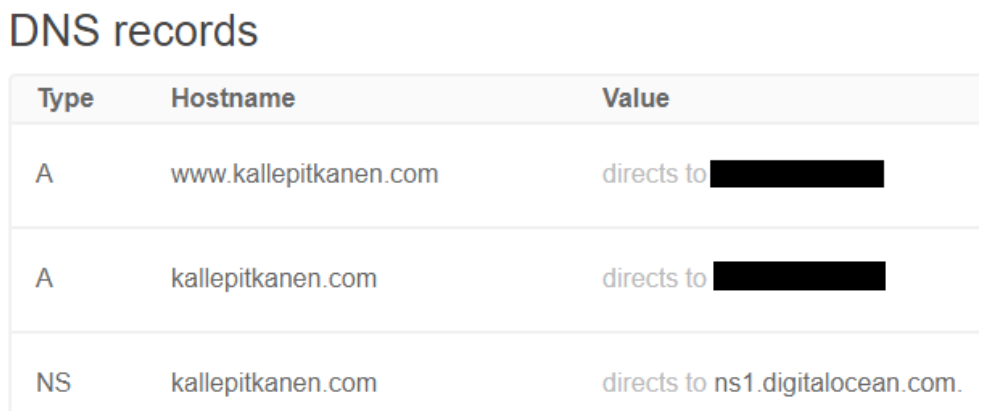
DigitalOceanin Dokku-dropletin pystytys oli muutaman painalluksen takana, aivan kuten DigitalOcean mainosti palvelua One-click-appina. Dokkun kanssa ei ollut yhtä helppoa kuin dokumentaatiossa, enkä saanut julkaisuketjua toimimaan nopeasti. Käytin noin päivän kokeillen kaikenlaisia eri ratkaisuja, mitä löysin hakukonehauilla. Lopulta ratkaisuksi ongelmaan selvisi, että Dokku ei löydä Dockerfileä jos se ei ole hakemistorakenteen juuressa. Ilmeisesti tätä tietoa pidettiin itsestäänselvyytenä, kun siitä ei kerrottu dokumentaatiossa. Päivitin toteutukseni hakemistorakenteen, jonka jälkeen julkaisuketju pyörähti onnistuneesti läpi.

Lopputulos oli erittäin kätevä, vaikka sen toteuttamiseen kului odotettua enemmän aikaa. Saan jatkossa julkaistua sovelluksen tuotantoon komennolla: ”git add . -A && git commit -m <commit message> && git push dokku” tai jos versionhallinnassa kaikki on ajantasalla, riittää ”git push dokku”. Komento päivittää Dokku-dropletille lisätyn versionhallinnan remote branchin eli etähaaran ja käynnistää Dockerin rakentamaan Docker imagea, joka asennetaan automaattisesti tuotantopalvelimelle. Koko julkaisuketjussa kestää noin 2-5 minuuttia.

3.16 Verkkotunnuksen ja IP-osoitteen yhdistäminen

Ensimmäisen onnistuneen julkaisun jälkeen verkkosivut olivat julkisesti kaikkien nähtävillä, mutta dropletin luoman IP-osoitteen takana. Osoitin NameCheapista tilaamani verkkotunnuksen NameCheapin nimipalvelimelta DigitalOceanille, jotta dropletin ja verkkotunnuksen hallinta olivat samassa paikassa. IP-osoitteen yhdistäminen verkkotunnukseen ta-

pahtui DigitalOceanin Networking-palvelussa, jossa IP-osoitteen ja verkkotunnuksen yhdistäminen toisiinsa tapahtui graafiselta käyttöliittymältä. Kuvassa 6 Type-sarakkeen A tarkoittaa tietuetta, joka osoittaa IP-osoitteen. NS tarkoittaa verkkotunnuksen nimipalvelinta.



Kuva 6. DigitalOceanin DNS records-käyttöliittymä

4 Yhteenveto

Opinnäytetyön viimeisessä luvussa arvioin omaa tekemistäni, tavoitteiden saavuttamista ja saavuttamaani lopputulosta.

Aloitin opinnäytetyön tekemisen kartoittamalla projektille potentiaaliset teknologiat. Ennen kartoitustyötä olin melko varma, että teen verkkosivut React.js:llä. Olin suunnitellut React.js-projektia jo pitkään ja nyt olisi ollut sille sopiva hetki. Onneksi kartoitusvaiheessa löysin Vue.js:n tarjoamat hyödyt, etenkin nopean oppimisen. React.js olisi ollut todennäköisesti liian hidas oppia tätä projektia varten, kun aika oli Vue.js:lläkin tiukoilla. React.js:ää olisi yhä hyödyllistä opetella, mutta toisaalta olisi ehkä tärkeämpää syventää osaamista Vue.js:llä.

Single-page applicationin toteuttaminen Vue.js:llä oli kaiken kaikkiaan todella opettavaista ja mielenkiintoista. Toteutus oli vielä melko yksinkertainen, joten en päässyt kunnolla hyödyntämään Vue.js:n tarjoamia ratkaisuja. Luultavasti tätä projektia kehitellessä eteenpäin, pääsen sukeltamaan vielä syvemmälle Vue.js:n tarjoamiin hyötyihin.

4.1 Haasteet

Dokumentaation puute tai vanhentuneet dokumentaatiot olivat projektin isoin haaste. Aikaa kului arviolta kaksi viikkoa pelkästään selvitellessä, miten jonkin asian kuuluisi toimia tai miksei asia toimi, kuten dokumentaatiossa on kuvattu. Syynä dokumentaation vajaavaisuuteen on todennäköisesti Vue.js kehittäjäyhteisön pieni koko verrattuna esimerkiksi React.js:ssään. Vaikka Vue.js on vain vuoden React.js:ää tuoreempi, se on kasvanut isoksi ja tunnetuksi paljon myöhemmin.

Vue.js:n nuoruus ilmeni myös paljon suppeammalla valikoimalla JavaScript-kirjastoja. Suosituimmatkin käyttämäni Vue.js:lle tehdyt JavaScript-kirjastot olivat pieniä verrattuina React.js:n vastaaviin. Toisaalta pääsin osallistumaan kehittäjäyhteisön toimintaan raportoimalla GitHubissa erilaisia ongelmia, ehdottamalla parannuksia ja kysymällä tarkennuksia dokumentaatioon.

Projektin toinen iso haaste oli aikataulu. Aloitin opinnäytetyöprojektin helmikuun puolivälissä ja sain päätökseen toukokuun alussa. Vajaa kolme kuukautta on lyhyt aika opetella yksin uusia toteutustapoja ja saada julkaistua valmis tuotos. Varsinkin, kun aikaa piti käyttää runsaasti myös opinnäytetyön kirjoittamiseen. Töiden ohella päivittäin käytettävissä ollut aikakin oli varsin pieni. Tein yleensä opinnäytetyöprojektia kello 18.30 – 23.00, tosin

loppuvaiheella illat venyivät hieman pidemmälle. Mennyttä kevättä voisi kuvailla erittäin koodausintensiiviseksi, kun olen töissäkin tehnyt paljon itselleni uusia asioita Knockout.js:llä ja C#:lla.

4.2 Tavoitteiden saavuttaminen

Tavoitteina oli suunnitella, toteuttaa ja julkaista toimeksiantajalle verkkoportfolio, joka on vähimmäisvaatimuksiltaan responsiivinen, näyttävä, toimiva ja tuo toimeksiantajan taideteokset hyvin esille. Verkkosivut ovat responsiiviset ja toimivat loistavasti resoluutioilla 2K näytöstä kaikenkokoisiin mobiililaitteisiin. 4K-resoluutio toimii hieman heikommin, mutta en usko kovin monen selaavan verkkoportfolioita televisiolla, joten 4K-resoluution tukeminen ei ole ensisijaista.

Toimeksiantajan kanssa olimme tyytyväisiä verkkosivujen ilmeeseen ja käytettävyyteen. Liitteissä näyttökuvat valmiista näkymistä. Valittu taustakuva toimii hyvin taustakuvana, sillä se tuo heti ilmi taitelijan persoonan ja tyylin. Käytettävyyden suhteen en ehtinyt toteuttamaan pientä parannusta näkymien välillä siirtymisessä. Olin asentanut toteutukseen Vue-fullpage.js-kirjaston, joka mahdollistaa näkymästä toiseen siirtymisen hiiren rullasta tai mobiililaitteella ruutua pyyhkäisemällä. Vue-fullpage.js-dokumentaatio oli puutteellista, eikä dokumentaatiosta löytynyt kuvausta, miten käyttöönotto tapahtuisi näkymien ollessa erillisiä komponentteja. Pieni animaatio näkymän vaihdossa ja helpompi näkymän vaihto olisivat tuoneet verkkosivuille hieman nykyaikaisempaa tuulahdusta. Palaan tähän jatkosuunnitelmat luvussa.

Totesimme toimeksiantajan kanssa, että Vue-Picture-Swipellä toteutettu kuvagalleria tuo taideteokset hyvin esille. Kuvakarusellin avaaminen on yksinkertaista, kuvasta toiseen siirtyminen on helppoa, kuvaa voi suurentaa saadakseen yksityiskohdat paremmin esille ja teokset voi jakaa sosiaalisessa mediassa. Näkyvyys sosiaalisessa mediassa on tietenkin hyväksi taiteilijalle.

Itselleni asettamani tavoitteet voisin sanoa ylittäneeni. Opin käyttämään itselleni uutta JavaScript Frameworkia ja sen kanssa työskentely tuntui projektin loppuvaiheessa varsin mukavalta. Sain julkaistua verkkoportfolion aikataulullaan ja julkaisuprosessin automatisoimisessa ylitin tavoitteet. Vähimmäistavoitteenani julkaisussa oli ainoastaan toimiva julkaisu. Julkaisua suunnitellessani ajattelin, että olisi helpointa toteuttaa tämä kunnolla alusta alkaen. Näinpä sain Dockerilla ja Dokkulla toteutettua julkaisun automaation yhteen lyhyeen komentoon.

4.3 Jatkokehityssuunnitelmat

Verkkoportfolion parantaminen jatkuu opinnäytetyön jälkeen. Parannan toteutustani mielelläni, sillä haluan oppia vielä lisää, enkä ole vielä täysin tyytyväinen lopputulokseen. Jatkamme siis toimeksiantajan kanssa yhteistyötä opinnäytetyön jälkeen. Seuraavina tavoitteina on helpottaa sisällönhallintaa, parantaa käyttöliittymää ja käytettävyyttä sekä mahdollisesti tuoda PWA täysiverisesti toteutukseen.

Ihan ensimmäiseksi teen loppuun muutamat pienet ”kesken” jääneet asiat. Nämä ovat vähimmäisvaatimusten ulkopuolella, joten sinänsä näillä ei ole kiirettä, mutta haluan saada valmiiksi aloittamani asiat. Vue-Fullpage:n näkymänvaihto-ominaisuuden selvittely ja käyttöönotto jäivät kesken. Jos Vue-Fullpage ei taivukkaan toteutukseeni, pitää toteuttaa jollain toisella tavalla näkymästä toiseen siirtyminen muutenkin kuin navigaation painikkeista.

Kirjoitushetkellä Vue-Picture-Swipe ei vielä tukenut kuvien kuvaustekstejä, mutta kävin asiasta keskustelua tekijän kanssa. Kuvaustekstien käyttöönotto jää toteutettavaksi heti kun tuki kuvausteksteille toteutetaan. Toisin sanoen Vue-Picture-Swipestä puuttui kyseiset propsit.

Edellisten jälkeen isompi kokonaisuus on toteuttaa sisällönhallintajärjestelmän käyttöönotto, jotta toimeksiantaja pystyisi itse ylläpitämään sisältöä. Siihen asti meillä on toimeksiantajan kanssa sovittuna, että toimeksiantaja toimittaa minulle päivitettävät sisällöt, jotka saan julkaistua nopeasti.

Hakukoneoptimointi on tärkeä asia, jota pitää vielä parantaa. Single-page application näyttää huonosti hakukoneboteille verkkosivujen sisältöä, kuten tärkeitä h1-otsikkotageja, tekstiä tai kuvia. Vue.js:lle on saatavilla lukuisia ratkaisuja näyttämään hakukoneboteille samat asiat, kuin mitä näkyisi perinteisillä MPA-sivuillakin.

PWA-toteutus olisi kirsikka kakun päälle, jotta verkkosivut toimisivat entistä paremmin mobiililaitteilla. PWA-toteutus on jo osittain valmis, sillä olen asentanut projektiin PWA:n vaatimat konfigurointitiedostot ja ne ovat jo osittain konfiguroituina. Vielä pitäisi viimeistellä konfiguraatiot ja saada SSL-sertifikaatti verkkotunnukselle, jotta verkkosivut saavat käyttöön PWA:n vaatiman HTTPS-protokollan.

4.4 Ammatillinen kasvu

Opinnäytetyön aikana vahvistin osaamistani kokonaisvaltaisesti web-kehityksessä. Taitoni ongelmanratkaisussa, suunnittelussa ja projektinhallinnassa paranivat. Erityisesti ammatilleni tärkeänä asiana osoitin pystyväni omaksumaan itselleni uuden web-teknologian suhteellisen nopeasti. Koska ohjelmistokehityskäytännöt ja ohjelmistokehitysteknologiat muuttuvat nopealla syklillä, on tärkeämpää pystyä oppimaan uutta, kuin olla jonkin sen hetkisen asian guru.

Onnistuin tekemään hyviä valintoja teknologioiden suhteen ja hallitsemalla hyvin kyseiset teknologiat, pystyisi varmasti muuntamaan osaamisen rahaksi. Vue.js oli mainio valinta tähän projektiin ja aion jatkossa kasvattaa Vue.js osaamistani, jotta pystyisin toteuttamaan monimutkaisempiakin toteutuksia Vue.js:llä. Vaikka opinnäytetyöprojektin ulospäin näkyvä lopputulos on verkkosivut, kyseessä on Vue.js:llä toteutettu verkkosovellus. Seuraavaksi haluaisin kokeilla Vue.js:llä backend-puolen tekemistä johonkin omaan testiprojektiini, eli hakea käyttöliittymään tietoa tietokannasta ja päivittää käyttöliittymältä tietoa tietokantaan.

Itselleni lähes uutena asiana opinnäytetyötä tehdessä tuli osallistuminen ohjelmistokehitysyhteisön toimintaan. Valtaosa ohjelmistokehityksen alustoista ja teknologioista kehittyvät avoimen lähdekoodin ansiosta. Avoin lähdekoodi mahdollistaa yhteisen agendan parantaa teknologioita yhdessä ja kehitellä uusia entistä parempia ratkaisuja. Oli hienoa päästä ehdottamaan uusia ominaisuuksia, päivityksiä dokumentaatioon ja raportoida kohtaamiani bugeja. Aion jatkossa kehittyä tällä saralla ja pyrkiä parhaani mukaan osallistumaan bugien ja parannusehdotusten selvittelyyn käyttämissäni avoimen lähdekoodin JavaScript-kirjastoissa. Yhteisön henkeen kuuluen pyrin pitämään toteutukseni avoimena lähdekoodina, mikäli se on projektin kannalta mahdollista.

Lähteet

Karol K. 2.6.2016. Best Practices for Background Images: Your “Getting Started” Guide. Luettavissa: https://theblog.adobe.com/best-practices-for-background-images-your-getting-started-guide/. Luettu 2.4.2018.

Aguinaga 2018. AngularJS is amazing... and hard as hell. Luettavissa: https://coderwall.com/p/3qclqg/angularjs-is-amazing-and-hard-as-hell. Luettu 17.3.2018.

Clifton 2003. What is a Framework? Luettavissa: https://www.codeproject.com/Articles/5381/What-Is-A-Framework. Luettu: 17.3.2018.

Cohan 2010. What is a Content Management System (CMS)? Luettavissa: http://www.comentum.com/what-is-cms-content-management-system.html. Luettu: 17.4.2018.

Cordle 2017. Why Angular 2/4 Is Too Little, Too Late. Luettavissa: https://medium.com/@chriscordle/why-angular-2-4-is-too-little-too-late-ea86d7fa0bae. Luettu: 17.3.2018.

Cromwell 2016. Evan You. Luettavissa: https://betweenthewires.org/2016/11/03/evan-you/. Luettu: 6.3.2018.

Eduonix 2017. An Infographic on Web Development Frameworks by Eduonix. Luettavissa: https://blog.eduonix.com/infographics/infographic-web-development-frameworks-eduonix/. Luettu 18.4.2018.

Elliot 2017. Top JavaScript Libraries & Tech to Learn in 2018. Luettavissa: https://medium.com/javascript-scene/top-javascript-libraries-tech-to-learn-in-2018-c38028e028e6. Luettu 17.3.2018.

Gil 2018. What Is a Domain Name? Luettavissa: https://www.lifewire.com/what-is-a-domain-name-2483189. Luettu: 12.2.2018.

Git 2018. Getting Started - About Version Control. Luettavissa: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control. Luettu: 2.4.2018.

Google 2018. Progressive Web Apps. Luettavissa: https://developers.google.com/web/progressive-web-apps/. Luettu: 14.3.2018.

Google 2018. Making AJAX applications crawlable. Luettavissa: https://developers.google.com/webmasters/ajax-crawling/docs/learn-more. Luettu: 13.3.2018.

Google Trends 2018. Luettavissa: https://support.google.com/trends/answer/4365533?hl=fi&ref_topic=6248052. Luettu: 26.2.2018.

Google Trends 2018. Luettavissa: https://trends.google.com/trends/explore?date=today%205-y&q=react.js,angular.js,vue.js. Luettu: 26.2.2018.

Json.org 2018. Luettavisa: https://www.json.org/. Luettu: 3.5.2018.

Laine 2015. Mitä markkinoijan tulee ymmärtää web-ohjelmoinnista. Luettavissa: https://www.dagmar.fi/verkkopalvelukehitys/mita-markkinoijan-tulee-ymmartaa-web-ohjelmoinnista/. Luettu: 22.2.2018.

Medium 2018. Top 32 Sites Built With ReactJS. Luettavissa: https://medium.com/@coderacademy/32-sites-built-with-reactjs-172e3a4bed81. Luettu: 6.3.2018.

Microsoft 2018. Model-View-Controller. Luettavissa: https://msdn.microsoft.com/en-us/library/ff649643.aspx. Luettu: 17.3.2018.

Muskardin 2018. Before You Build a PWA You Need a SPA. Luettavissa: https://hackernoon.com/before-you-build-a-pwa-you-need-a-spa-e22770a0f31c. Luettu: 14.3.2018.

NameCheap 2018. What is WhoIsGuard? Luettavissa: https://www.namecheap.com/support/knowledgebase/article.aspx/278/37/what-is-whoisguard. Luettu: 31.3.2018.

Peters 2012. 3 Key Software Principles You Must Understand. Luettavissa: https://code.tutsplus.com/tutorials/3-key-software-principles-you-must-understand--net-25161. Luettu 25.4.2018.

Pääkkö 2017. Progressive Web App, mistä oikein on kyse? Luettavissa: https://www.symbio.com/fi/progressive-web-app-mista-oikein-kyse/. Luettu: 13.3.2018.

Reactjs 2018. Tutorial: Intro To React. Luettavissa: https://reactjs.org/tutorial/tutorial.html. Luettu: 17.3.2018.

Saring 2018. 11 Vue.js UI Component Libraries You Should Know In 2018. Luettavissa: https://blog.bitsrc.io/11-vue-js-component-libraries-you-should-know-in-2018-3d35ad0ae37f. Luettu: 1.4.2018.

Schneider 2018. Single-Page vs. Multi-page UI Design: Pros & Cons. Luettavissa: https://www.uxpin.com/studio/blog/single-page-vs-multi-page-ui-design-pros-cons/. Luettu: 13.3.2018.

Scott & Mondago 2018. 6 Reasons for Employing Component-based UI Development. Luettavissa: https://www.tandemseven.com/technology/6-reasons-component-based-ui-development/. Luettu: 1.4.2018.

Skólski 2018. Single-page application vs. multiple-page application. Luettavissa: https://neoteric.eu/single-page-application-vs-multiple-page-application?utm_source=medium.com&utm_medium=social&utm_content=neo&utm_campaign=blog. Luettu: 13.3.2018.

Stack Overflow Insights 2018. Luettavissa: https://insights.stackoverflow.com/. Luettu: 26.2.2018.

Stack Overflow Trends 2018a. Luettavissa: https://insights.stackoverflow.com/trends?tags=jquery%2Cangularjs%2Cangular%2Creactjs. Luettu: 26.2.2018.

Stack Overflow Trends 2018b. Luettavissa: https://insights.stackoverflow.com/trends?tags=meteor%2Cbackbone.js%2Cember.js%2Cvue.js%2Cknockout.js%2Credux. Luettu: 26.2.2018.

Storyblok 2018. Flexible pricing. Luettavissa: https://www.storyblok.com/pricing. Luettu: 3.5.2018.

Vase 25.2.2018. Docker ja konttiteknologiat – ratkaisuja ja suorituskykyä. Luettavissa: http://eu.landisgyr.com/better-tech/docker-ja-konttiteknologiat-ratkaisuja-ja-suoritus-kyky%C3%A4. Luettu: 6.5.2018.

Vaughan-Nichols 2018. What is Docker and why is it so darn popular? Luettavissa: https://www.zdnet.com/article/what-is-docker-and-why-is-it-so-darn-popular/. Luettu: 4.5.2018.

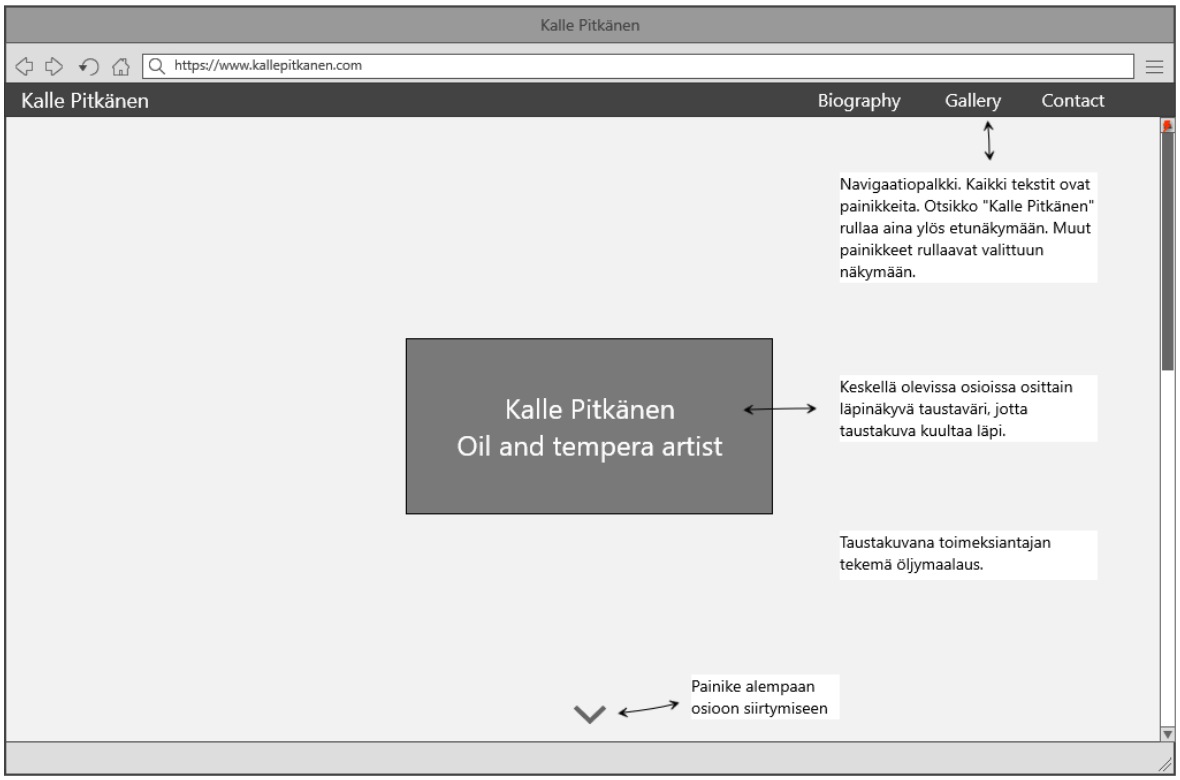
Vuejs 2018. Building Large-Scale Apps. Luettavissa: https://v1.vuejs.org/guide/application.html. Luettu: 18.3.2017.

Vuetify 2018. Grid system. Luettavissa: https://vuetifyjs.com/en/layout/grid. Luettu: 17.4.2018.

W3Schools 2018. CSS Flexbox. Luettavissa: https://www.w3schools.com/css/css3_flexbox.asp. Luettu: 17.4.2018.

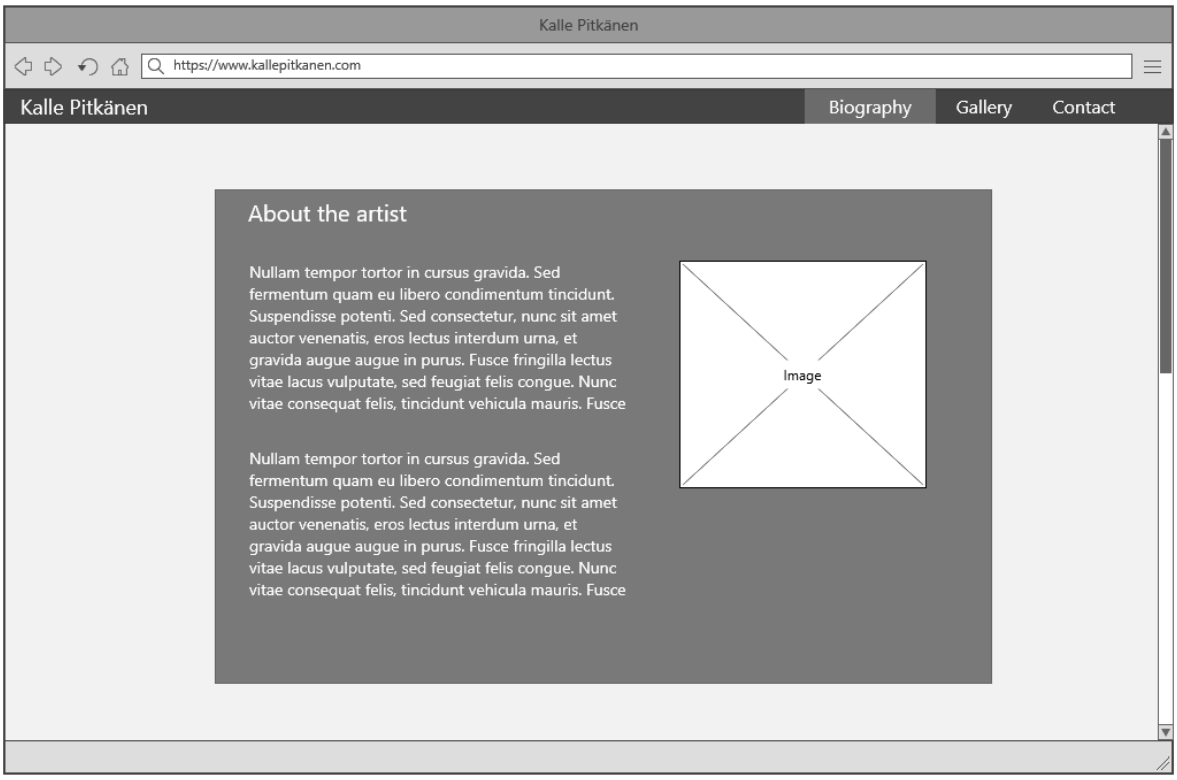
Wordpress 2018. Trusted by the Best. Luettavissa: https://wordpress.org/. Luettu: 17.4.2018.

Liitteet

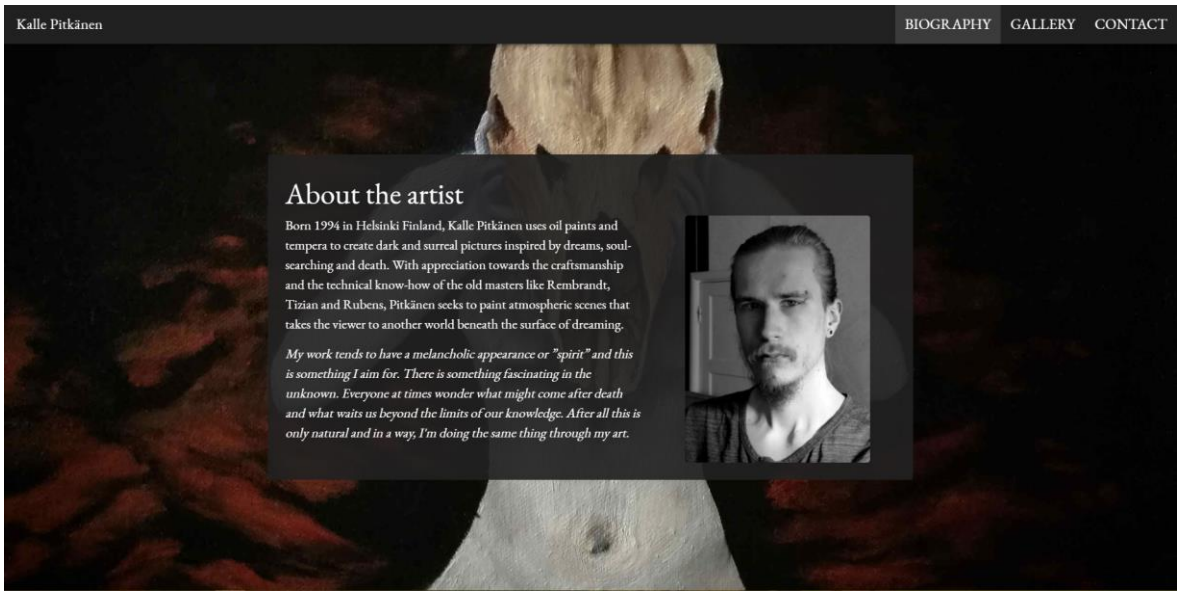


Kuva 7. Main page -näkymä käyttöliittymäsuunnitelma.

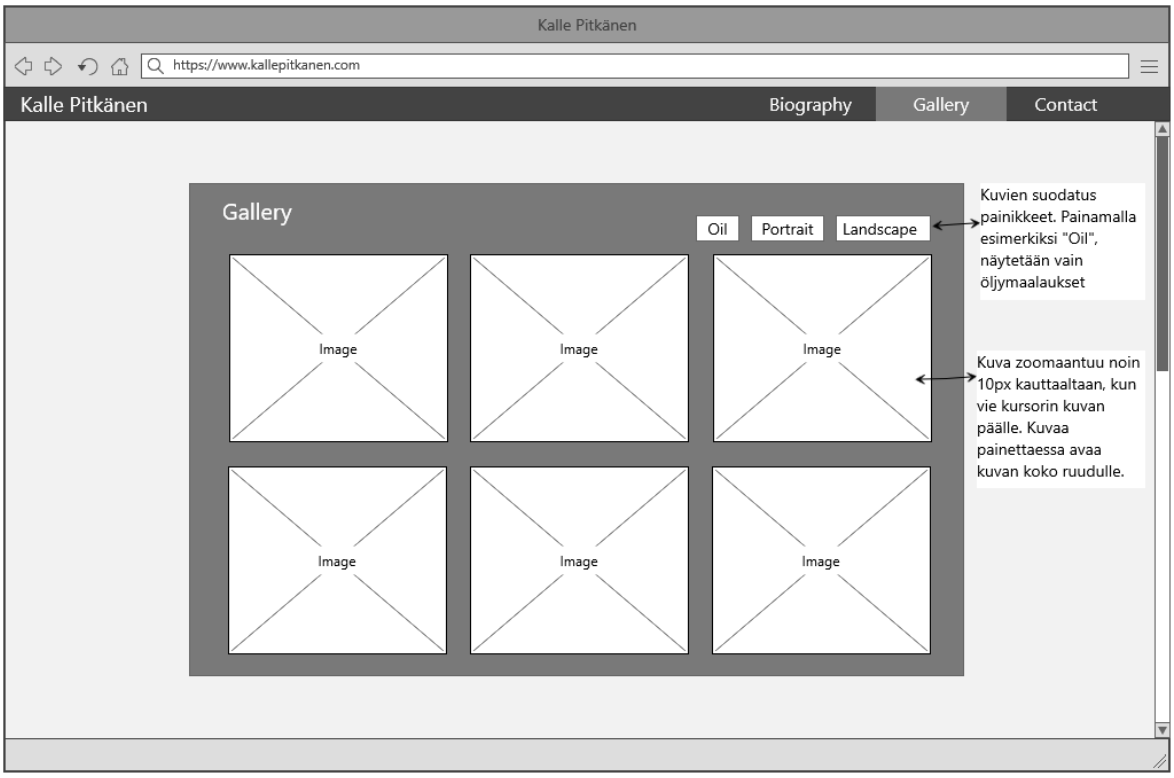
Kuva 8. Main page -näkymä toteutunut.



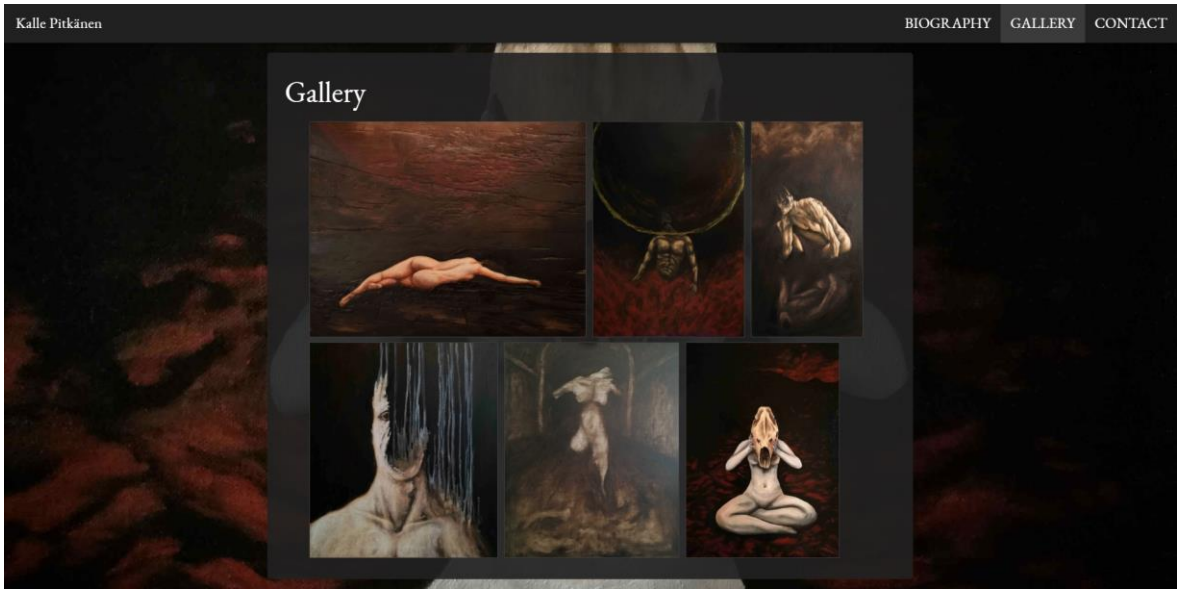
Kuva 9. Biography-näkymä käyttöliittymäsuunnitelma



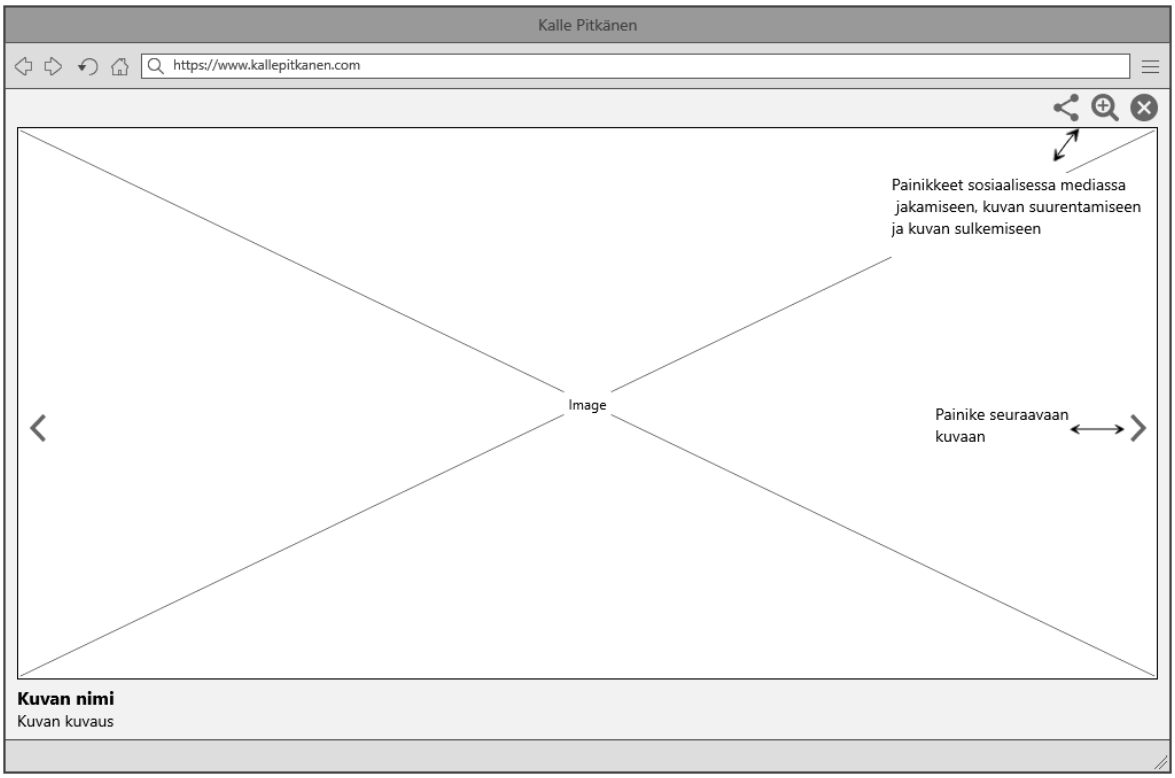
Kuva 10. Biography-näkymä toteutunut



Kuva 11. Gallery-näkymä käyttöliittymäsuunnitelma

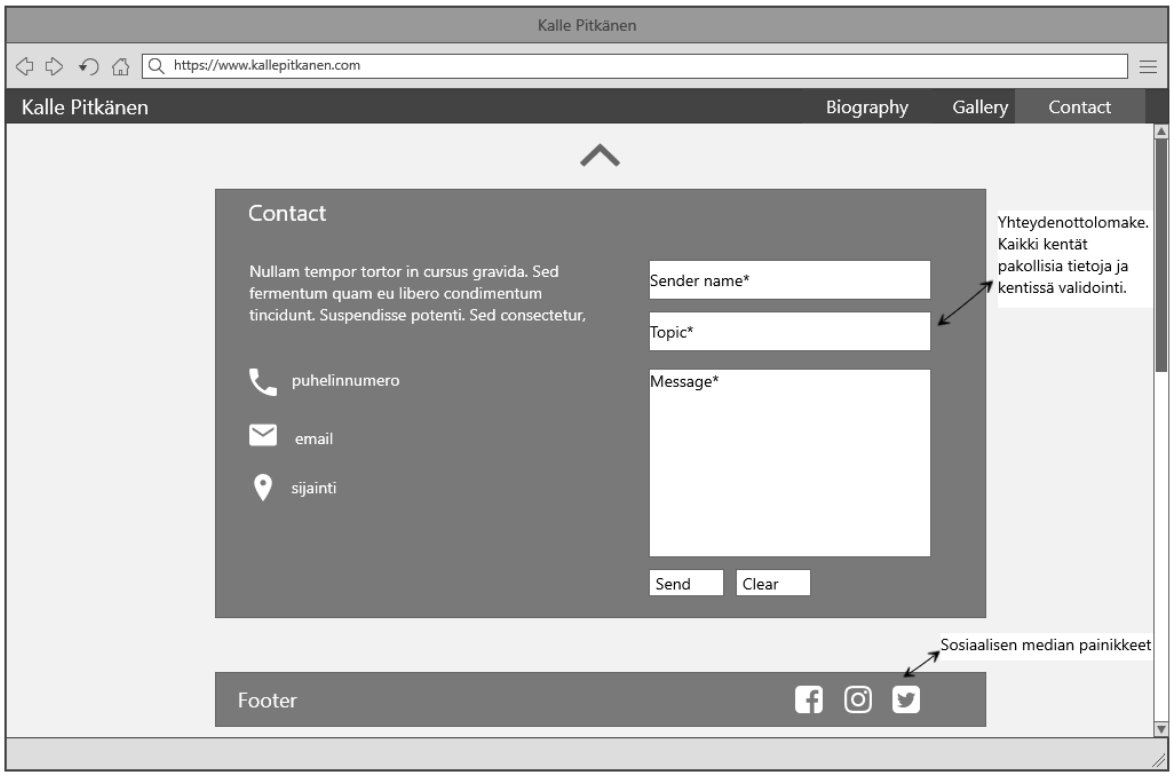


Kuva 12. Gallery-näkymä toteutunut

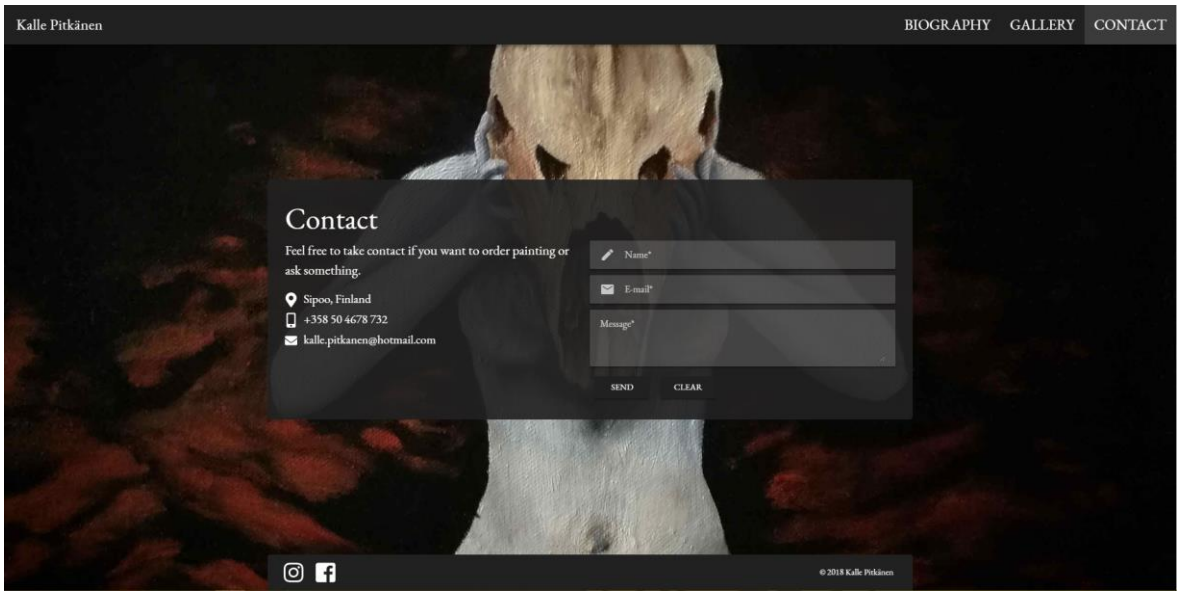


Kuva 13. Picture-näkymä käyttöliittymäsuunnitelma

Kuva 14. Picture-näkymä toteutunut



Kuva 15. Contact-näkymä ja footer käyttöliittymäsuunnitelma



Kuva 16. Contact-näkymä toteutunut